

## **Petit Computer Help**

2012 SmileBoom Co.,Ltd  
All rights reserved

## **1. Introduction**

### **1-1 About Petit Computer**

This software is an easy-to-use BASIC programming language tool allowing you to write your own BASIC programs. By utilizing DS Wireless Communications, you can share your programs with friends, and receive images created by them. This will allow you to collaborate to create programs.

### **1-2 User Agreement**

- Programs and graphics created using this software can be shared with large numbers of users. Please refrain from creating any content other users may find offensive, or any content that might reveal personal information about yourself or others, or violate any other party's intellectual property rights.
- You risk prosecution if any programs you create cause offence, are obscene, or are libelous.
- Gamebridge accept no liability for any adverse consequences of any programs or images users choose to share.

### **1-3 Program Precautions**

- The BASIC language used in this software is not compatible with pre-existing versions of BASIC. Please be aware of the differences in the programming languages when attempting to port over older BASIC programs.
- This software uses fixed-point number representation, which can have large margins of error for certain calculations, resulting in precision loss. This means it should not be used for programs which require precise calculations.
- This software does not use branch instructions giving line numbers. Branch instructions use the @ mark and the line tag for specific lines.
- Complex programs with increasing amounts of visual data can lead to a reduction in processing speed.
- If commands that write to files such as SAVE, RECVFILE and DELETE are repeatedly used, it may take longer to save or load these files.
- If you enter reserved commands in lower case letters, they will automatically be converted to upper case.
- In this software, comparisons are represented using == for 'equals' and != for 'differs'. (Similar to the C programming language)
- When FOR commands are entered in this software, the conditions are determined first. For a case like FOR I=0 TO -1 where STEP1 will not function, it

will skip the FOR command and carry on running the commands after NEXT. Unlike existing BASIC programs, it is not guaranteed to carry out the command once.

- This software's program text editing function allows you to store a maximum of 100 characters per line, with a maximum of 9999 lines. However, when the memory allocated to saving text is exceeded, you will no longer be able to enter text. The maximum number of characters you can use in a program is approximately 520,000.
- Occasionally, graphics which you have included in your program are not properly displayed on screen. This may be caused by a variety of factors, such as the previously run program specifying that the display should be on the Lower Screen. If this should occur, switch to Run Mode and run the following routine:

[ACLS \(Enter\)](#)

If your graphics are still not displayed, it may be because the color selected is not visible, or the graphics are made up of transparent characters. In this case, use the COLINIT or CHRINIT commands to reset the display.

- If the numbers you wish to use when performing divisions or other calculations are all integers, use the FLOOR command. The calculation will then only be performed using integers. When calculating coordinates, calculation errors may combine to give slight inaccuracies.
- Keyboard input is detected via INKEY\$(). However, the system requirements mean that some keys will not be detected,
- Label names can be replaced by character string variables for a certain set of commands. But this can only be done for those commands with which labels can be used.
- A maximum of 16 channels can be used simultaneously for background music, sound effects.

#### 1-4 Controls

This software is a development tool, so its controls differ from those of normal games.

<b>+Control Pad</b>	Moves cursor in Edit Mode. (Press up and down while holding down the L Button to scroll up and down one page at a time, and
---------------------	---

	press left and right to go to the start and end of a line.)
A,B,Y,X Button	Can be used while a program is running. The Y Button functions as backspace when in Edit Mode, while A functions as the Enter key.
L,R Button	This function like the Shift key on a keyboard, letting you input the purple characters on the keyboard. It can be used while a program is running.
START Button	If you have a program saved, press START to run it. This can be used while a program is running.
SELECT Button	SELECT functions like the ESC key when programs are running.

In Run Mode, the screen display can be reset by simultaneously pressing left on the + Control Pad, the R Button, and START. This also restores the font to its default, making it useful after a game program when elements of the display are difficult to view.

#### 1-5 What is a Program?

Computers are marvelous machines that can do all manner of things, from following people's instructions and displaying certain images on screen, to playing sounds or detecting what is inputted via the Touch Screen.

A program is a set of instructions that tells the computer precisely what to do. Petit Computer uses a language called BASIC, and its structure is close to the grammar we use every day to communicate.

For example:

- Display text on screen with a PRINT command
- Play a sound with a BEEP command
- Draw a circle on screen with a GCIRCLE command
- Display a character with an SPSET command
- Store a certain number in the memory by
- Assigning a variable
- Find out the value of a number using
- Comparison and branch commands
- Save your program with a SAVE command

BASIC has a huge range of instructions to get the computer to do what you want it to. While it would be hard work to remember every single command, if you try out different commands, you'll learn them little by little and be able to program games and tools.

It may be challenging at first, but this software is all you need to create and run your very own programs. Of course, sometimes you might enter the wrong command by mistake, creating a bug.

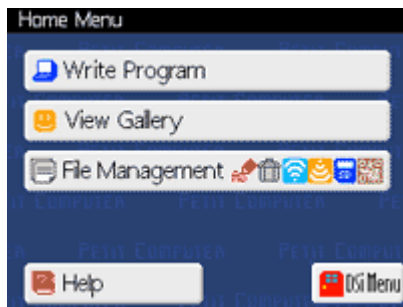
Many of the programmers who create video games got started by using BASIC. If you've set your sights on developing games in the future, this is a great way to experience the fun of programming first-hand.

#### 1-6 Glossary

Term	Info
Characters	The 8x8 pixel 16-color images used to make up sprites and BG screens.
Color	There are 256 colors available for graphics, sprites and BG (background). (0 is transparent)
Palette	The range of colors that can be used for graphics and characters. 1 palette= 16 colors (0 is transparent)
Background(BG)	A display function where the screen is covered in characters, used for example on map screens.
Sprite	A function that can display up to 100 animated characters.
System Icons	Icon buttons used to switch between modes and tools.
Pixel	The colored dots that make up the images and characters on screen.
Frame	A unit for determining length an image is displayed for. 1 frame = 1/60th sec
VSYNC	This is function that refreshes the screen display. The smallest unit of time for the screen to be refreshed is a single frame.
File	The memory assigned to saving programs and images you have created.
Resources	The general name used in BASIC for images and programs read

	from files. Some are saved in the internal memory, while others are saved in the video memory.
Track	This is the source from where background music is played. Music can be played using a maximum of 8 tracks simultaneously.
Channel	This is the unit in which MML music is played. A maximum of 8 types of instrument can be played at the same time.
MML	Music Macro Language This refers to the command strings used create music.
Start Point x Start Pont y	The coordinates (X=horizontal, y=vertical) for the top left point of a rectangular area.
End Pont x End Point y	The coordinates (X=horizontal, y=vertical) for the bottom right point of a rectangular area.
Transfer Destination x Transfer Destination y	The coordinates (X=horizontal, y=vertical) for the top left point of the rectangular area of the transfer destination.

## 2. Home Menu



### 2-1 About the Home Menu

A range of options for Petit Computer are accessible from this menu, and you will return here after programs end. Select from the following 5 options:

#### 2-1-1 Write Program

Launch Petit Computer. Touch this button when you want to write programs.

#### 2-1-2 View Gallery

Select a program you wish to run from a list of saved programs.

#### 2-1-3 File Management

Copy, rename or delete saved files, or transmit and receive data.

#### 2-1-4 Help

View the instruction manual for Petit Computer as well as a list of BASIC commands.

#### 2-1-5 DSi Menu

Exit Petit Computer and return to the DSi menu.

### 2-2 Create Program

See “Create Program (detail)” to learn how to use Petit Computer.

### 2-3 View Programs



This lets you select and run programs and samples, Touching the Run button will launch Petit Computer and run the program.

- Reached end of program
- Error in program
- ESC key or Cancel button pressed
- SELECT pressed
- END command run in program

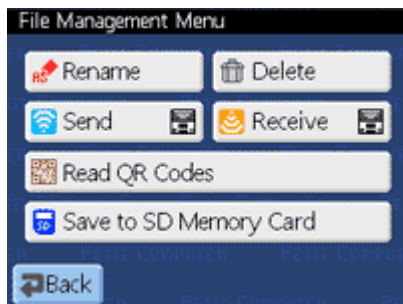
In any of the cases listed above, the program that is currently running will be cancelled you will exit Petit Computer and return to the Home Menu. All current variable data will be lost when the program finishes. Utilize the SAVE command to store data if required. When running programs in this mode, use the Edit button to view the program list.

## 2-4 File Management

See “File Management (detail)” for more information on managing files.



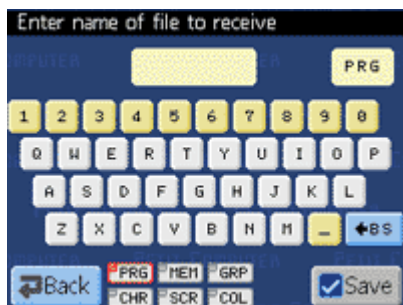
### 3 File Management(detail)



#### 3-1 Change Name

This function allows you to rename your saved files by following the instructions below:

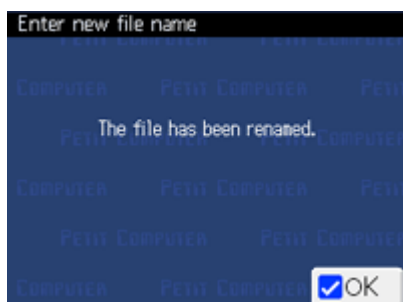
##### 3-1-1 Select file you wish to rename.



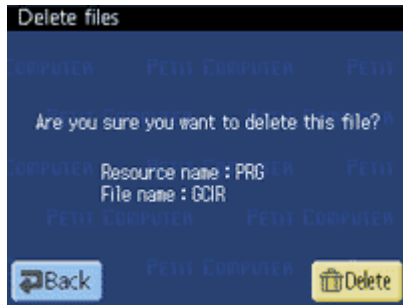
##### 3-1-2 Change the name using the simplified keyboard.



##### 3-1-3 The file will now have been renamed.



#### 3-2 Delete File



This function allows you to delete saved files by following the instructions below:

3-2-1 Select the file you wish to delete.

3-2-2 Confirm that you wish to delete the selected file.

3-2-3 The file will now have been deleted.

### 3-3 Send File

This function allows you to send saved files to other users by following the instructions below:

3-3-1 Select the file you wish to send.

3-3-2 Dialog box will appear.

(The operation is then identical to that performed by the SENDFILE command.)

### 3-4 Receive File

This function allows you to receive files from other users by following the instructions below:

3-4-1 Enter the name of the file you wish to receive.



3-4-2 Dialog box will appear.

(The operation is then identical to that performed by the RECVFILE command.)

### 3-5 Read QR Code

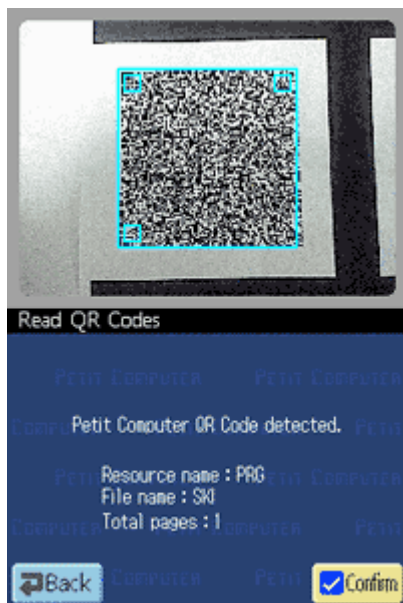
While reading QR codes, the system's processing power is focused on this operation, so scrolling may stop and buttons become temporarily less responsive.

This function allows you to use the external camera to read special QR codes for use with Petit Computer. To read QR codes and create files, follow the instructions below:

#### 3-5-1 Line up the first QR code so that it is within the red frame



#### 3-5-2 When the QR Code is detected, the number of pages will be displayed.



#### 3-5-3 The required number of pages will be read.

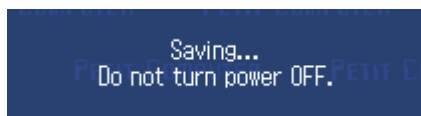
3-5-4 After the pages are read, the file will be created."

### 3-6 Save to SD Memory Card

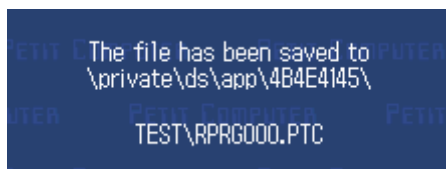
"This function allows you to copy saved files onto an SD Memory Card by following the instructions below:

#### 3-6-1 Select the file you wish to write

#### 3-6-2 The writing process will begin



#### 3-6-3 The selected file will be written onto the SD Memory Card



¥private¥ds¥app¥4B4E414A¥File Name

Once the file has been successfully written to the SD Memory Card, the folder above will be created and the administrative file will be accessed.

<Please Note When Transferring Files>

**Not enough available space on Memory Card/Write-Protect switch ON:**

In these cases, it will not be possible to transfer data onto the SD Memory Card. Please ensure that there is enough space and that the Write-Protect switch on your SD Memory Card is OFF.

The properties of files on SD Memory Cards are not accessible, meaning that users will be unable to view details of the files. Please refrain from trying to analyze or modify file contents.

### 3-7 Create QR Code Tool

You cannot retrieve files saved to SD Memory Cards, but file contents can be viewed and QR Codes created by making use of a free online application via your web browser. For more details, please see the official Petit Computer homepage.

4 Create Program (detail)

Touch the 'Create Program' button on the Home Menu to launch SmileBASIC.  
SmileBASIC has the following three modes:

●Run Mode

You can enter directly executable commands using the keyboard. (Switch from Edit Mode with the Run button.)



●Edit Mode

An editing function that allows you to write program source code. (Switch from Run Mode with the Edit button.)



●Help

See instructions for programming and a list of BASIC commands. (Switch with Manual button.)



4-1 Switching Modes

Use the System Icons to switch modes.

System Icons (in Run Mode)

1	Manual Icon This explains how to use this software.
2	Run Mode Icon

	Switches to Run Mode.
3	Edit Mode Icon Switches to the Edit Mode to edit text.
4	User System Icon Area Can be used within programs.

### System Icons (in Edit Mode)

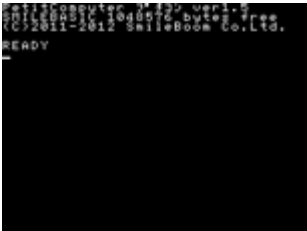
5	Copy Copy the line at current cursor position.
6	Paste Insert a copied line into the program at current cursor position.

### 4-2 Run Mode

This mode allows you to run programs you have written. To run programs written in Edit Mode, enter RUN using the keyboard and touch the Enter key. In Run Mode you can also save and load programs.

### Upper Screen: Console

Commands entered using the keyboards are displayed here.



### Lower Screen: Keyboard display



Input Switch Button	
A	ABC
♥	Symbols
ｱ	Kana

Touching the Input Switch button will change the characters available on the keyboard. Pressing SHIFT will enable you to input further characters. In Run Mode, PNLTYPE will not switch the keyboard.

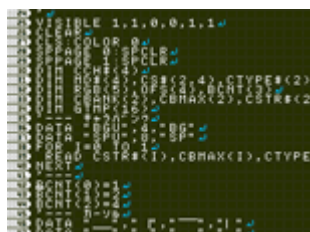
"Delete files saved using the SAVE command or change file names in Run Mode.  
For more details, see 12 ©Run Mode Commands."

### 4-3 Edit Mode

Write your own programs in Edit Mode. The varieties of commands at your disposal give you plenty of scope to create programs in your own way. To see the commands you can use, open this manual and refer to the sections on commands and functions.

- Upper Screen: Source display

Displays strings entered using the keyboard. Line numbers are automatically added when page breaks are inserted. Up to 100 characters per line can be displayed on screen. If the line is too long to be displayed, the line will scroll horizontally to display the other characters.



- Lower Screen: Keyboard display



As in Run Mode, touching the Input Switch button will change the keyboard.

## 4-4 Command Entry Support

If you forget commands when writing a program, you can enter letters and numbers to search for a list of suggested commands and functions.

## 1) Suggestion List

Choose from suggested commands and functions in the white area below the function keys on the lower screen.



## 2) Initial Suggestion Screen

Press the G key on the keyboard and all the commands and functions beginning with that letter will be listed.



## 3) Narrowing Down the List

If you then press the C key on the keyboard, the list of suggestions will be narrowed down to those beginning with GC.



## 4) Selecting a Suggestion

If you find the term you are looking for, touch it and it will be entered in your program. This reduces the time it takes to enter a command such as GCIRCLE.

>> << When there are too many terms to fit in the Suggestion List, press this button to go to the next page.

## 4-5 Searching (in Edit Mode)

Touch the magnifying glass icon at the bottom right of the keyboard in Edit Mode and you will be able to enter a search term below the function keys. This is useful when you are writing a long program and wish to quickly search for labels and function names.



Enter the search term and press ENTER to begin the search. By pressing up or down on the +Control Pad, you can indicate whether to search above or below the current line in the program. Press the magnifying glass icon again to exit Search Mode.

## 4-6 Writing Simple Programs

Here's a step-by-step guide to writing a program that will display strings on the



console screen and play sound. After starting Petit Computer, execute the following commands, ensuring you have not run any sample programs beforehand. (When there are no programs, you will always start in Run Mode )

- 1) Touch the Edit button to enter Edit Mode.
- 2) Enter PRINT ""WORDS"" in the first line.
- 3) Enter BEEP in the second line.

```
0001 PRINT ""WORDS""
```

```
0002 BEEP
```

- 4) Touch the Run button to enter Run Mode.
- 5) Input RUN (Enter).

WORDS

OK

Text will be displayed on screen, and a sound will be played. It is also possible to run commands directly in Run Mode.

```
GLINE 0,0,255,191,15 (Enter)
```

This command will draw a diagonal line from the top left of the upper screen to the bottom right.

```
GCLS (Enter)
```

This command will remove the line without affecting the text. When you're experimenting with different commands to find out precisely what they do, it's a good idea to do it directly in Run Mode.

## 5 Sample File

This software includes the following sample programs. For more details, see the programming code for each.

### 5-1 Using Sample Files

- 1) Go to Run Mode
- 2) EXEC"SAMPLE2" (Press Enter)
- 3) Press SELECT to pause
- 4) View program in Edit Mode

By entering other sample names instead of the SAMPLE2 contained within the "", you can check out other sample programs. You are also free to save any of the samples included in the software under different names. You can then tweak them to your heart's content without needing approval from SmileBoom first.

### 5-2 Basic Samples

These are simple programs designed to teach users how to use the standard commands in BASIC. You can view the programs in Edit Mode and use them as a reference when writing your own BASIC programs.

File Name	Description
SAMPE1	Displays characters on the console.
SAMPLE2	A simple character-input calculator.
SAMPLE3	A simple piano program using the keyboard.
SAMPLE4	A number-guessing game.
SAMPLE5	A bio-rhythm program.
SAMPLE6	A lower screen sequencer.
SAMPLE7	A shooting game with moving foes.
SAMPLE8	A program demonstrating useful commands that control variables.
SAMPLE9	A demo using sprites and background (BG).
SAMPLE10	A demo using multi-page graphic screens.
SAMPLE11	A program demonstrating music created and played using MML(Music Macro Language).
SAMPLE12	A program showing you how to create you own unique instrumental sounds.

### 5-3 Advanced Samples

These are advanced programs using a wide variety of BASIC commands. Each of these samples can also be used when you make your own games. You can view the programs in Edit Mode, and once you have mastered the BASIC commands, you can add your own unique features.

File Name	Description
CHRED	A character creation tool.
SCRED	A background screen creation tool.
GRPED	A 256-color graphic tool.
DRWED	A drawing tool that stores elements in units such as lines and squares.
GAME1	A racing game where you erase on-screen dots.
GAME2	A dungeon-crawling RPG.
GAME3	Blast intergalactic foes in a vertically-scrolling shoot-em-up.
GAME4	A shooting game where you dodge a barrage of bullets while taking out enemies.
GAME5	A fighting game where giant characters do battle.

### 5-4 Character Creation Tool

EXEC "CHRED" (Press Enter to Run)



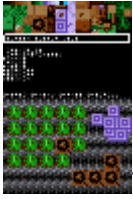
This tool can be used to create character data for backgrounds or sprites.

Select color samples and tools and input them in the Edit Area. Press A Button to access File Mode, enter L (or S) and then press ENTER to load or save. You can access and use saved data via programs.

(ex.) LOAD "BGU0:MYBG00"

### 5-5 BG Screen Creation Tool

EXEC "SCRED" (Press Enter to Run)

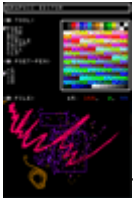


This is a tool for creating BG screen data, for example, creating rows of houses. Select a character from the samples to paste them to the Edit Area. You can access and use saved data via programs.

(ex.) LOAD "SCU0:MYSC00"

#### 5-6 Graphic Creation Tool

EXEC "GRPED" (Press Enter to Run)



This is a tool for creating graphic data for the graphics Screen, utilizing up to 256 colors. Select a color from the samples and draw a picture in the Edit Area. You can access and use saved data via programs.

(ex.) LOAD "GRP0:MYGRP00"

#### 5-7 DRWED (Drawing Pictures)

EXEC "DRWED" (Press Enter to Run)



This drawing tool can store up to 8000 commands for drawing elements such as lines and squares. It differs from pixel art by allowing you to replay stored data and adjust the picture as many times as you like. The picture data is stored on the upper screen in the form of colors.

To learn more about how this tool recreates pictures using stored color data, view its program in Edit Mode. You can take the elements used to recreate images and use these in your own programs.



## 6 Sample Games

These are sample games that have been created using Petit Computer. You can view the code for these games in Edit Mode. Feel free to adapt the program to your own needs, making enemies weaker, for example, or adjusting the speed of bullets.

### 6-1 GAME 1 (Road Racer)

EXEC "GAME1" (Press Enter to Run)

Steer your car with the +Control Pad, aiming to pass over all the dots on the screen. Your car will accelerate as it passes over dots.



### 6-2 GAME 2 (3D Dungeon Crawler)

EXEC "GAME2" (Press Enter to Run)

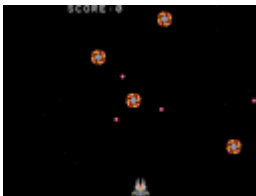
In this RPG, you'll use the +Control Pad to head to the top of the dungeon while bashing beasts and picking up items such as medicine and weapons.



### 6-3 GAME 3 (Side-On Shooter)

EXEC "GAME3" (Press Enter to Run)

Use the +Control Pad to move and the A Button to shoot as you zap the incoming enemies and take aim at the big boss.



### 6-4 GAME 4 (Barrage Blaster)

EXEC "GAME4" (Press Enter to Run)

Advanced sprite commands fill the screen with foes and firepower. Avoid enemy strikes, blast your enemies and take on the giant boss. The scrolling background has

three layers.



6-5 GAME 5 (Sword Fight)

EXEC "GAME5" (Press Enter to Run)

Two vast screen-filling characters wield mighty swords as they duel to the death.



## 7 SPRITE

### 7-1 SPU0 - 1(0-127)

SPU0: Standard Game Elements



SPU1: Young Boy & Sorceress



### 7-2 SPU2 - 3(128-255)

SPU2: Smaller Enemies etc.





SPU3: Effects, Items etc.



7-3 SPU4 – 5(256-383)

SPU4: Flying Objects



SPU5: Larger Enemies, Explosions etc.



7-4 SPU6 – 7(384-511)

SPU6: Larger Horizontal Enemies

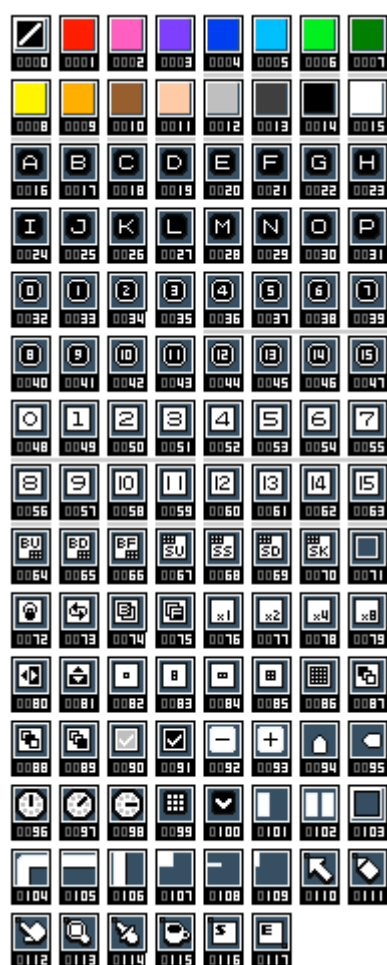


SPU7: Battleship



7-5 SPS

Standard Characters for Lower Screen



## 7-6 System Icons

Standard Icons for Tools etc.



## **8 Character List (Background)**

8-1 BGU0(0-255)

00	01	02	03	04	05	06	07
08	09	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103
104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183
184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231
232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247
248	249	250	251	252	253	254	255

8-2 BGU1(256-511)

256	257	258	259	260	261	262	263
264	265	266	267	268	269	270	271
272	273	274	275	276	277	278	279
280	281	282	283	284	285	286	287
288	289	290	291	292	293	294	295
296	297	298	299	300	301	302	303
304	305	306	307	308	309	310	311
312	313	314	315	316	317	318	319
320	321	322	323	324	325	326	327
328	329	330	331	332	333	334	335
336	337	338	339	340	341	342	343
344	345	346	347	348	349	350	351
352	353	354	355	356	357	358	359
360	361	362	363	364	365	366	367
368	369	370	371	372	373	374	375
376	377	378	379	380	381	382	383
384	385	386	387	388	389	390	391
392	393	394	395	396	397	398	399
400	401	402	403	404	405	406	407
408	409	410	411	412	413	414	415
416	417	418	419	420	421	422	423
424	425	426	427	428	429	430	431
432	433	434	435	436	437	438	439
440	441	442	443	444	445	446	447
448	449	450	451	452	453	454	455
456	457	458	459	460	461	462	463
464	465	466	467	468	469	470	471
472	473	474	475	476	477	478	479
480	481	482	483	484	485	486	487
488	489	490	491	492	493	494	495
496	497	498	499	500	501	502	503
504	505	506	507	508	509	510	511

8-3 BGU2(512-767)



8-4 BGU3(768-1023)

768	769	770	771	772	773	774	775
776	777	778	779	780	781	782	783
784	785	786	787	788	789	790	791
792	793	794	795	796	797	798	799
800	801	802	803	804	805	806	807
808	809	810	811	812	813	814	815
816	817	818	819	820	821	822	823
824	825	826	827	828	829	830	831
832	833	834	835	836	837	838	839
840	841	842	843	844	845	846	847
848	849	850	851	852	853	854	855
856	857	858	859	860	861	862	863
864	865	866	867	868	869	870	871
872	873	874	875	876	877	878	879
880	881	882	883	884	885	886	887
888	889	890	891	892	893	894	895
896	897	898	899	900	901	902	903
904	905	906	907	908	909	910	911
912	913	914	915	916	917	918	919
920	921	922	923	924	925	926	927
928	929	930	931	932	933	934	935
936	937	938	939	940	941	942	943
944	945	946	947	948	949	950	951
952	953	954	955	956	957	958	959
960	961	962	963	964	965	966	967
968	969	970	971	972	973	974	975
976	977	978	979	980	981	982	983
984	985	986	987	988	989	990	991
992	993	994	995	996	997	998	999
1000	1001	1002	1003	1004	1005	1006	1007
1008	1009	1010	1011	1012	1013	1014	1015
1016	1017	1018	1019	1020	1021	1022	1023





## 9 Sound-Related Table

Preset commands and other items related to background music, BEEP and TALK are gathered in one table.

### 9-1 Preset Music

No.	Info
0	Jolly and Jaunty
1	Dark and Dank
2	Tension is Rising
3	Upbeat Emotion
4	Opening Jingle
5	Clear Jingle
6	Game Over
7	Menu Select
8	Result Screen
9	Staff List
10	Staff List 2
11	Classical Drama
12	Marching Band
13	Ultra-hard Rock
14	Jolly and Jaunty 2
15	WOND
16	Deep in Thought
17	WOND2
18	For the Future
19	BAL
20	BAL_2
21	Espionage
22	SCI
23	Shooting Song
24	Pad
25	SEN
26	Pure
27	ROA
28	CUA

29	FIG
----	-----

## 9-2 Preset Sound Effects

No.	Info
0	Beep
1	Noise
2	Cursor Movement
3	Confirm
4	Cancel
5	Ascend
6	Descend
7	Coin
8	Jump
9	Land
10	Fire
11	Damage
12	Metal
13	Explosion
14	Scream
15	Brake
16	Banjo
17	Synth Strings
18	Synth Brass
19	Synth Bass
20	Guitar
21	Organ
22	Piano
23	Cow Bell
24	Tom-Toms
25	Cymbals
26	
27	ROA
28	CUA
29	FIG
30	Snare Drum

31	Bass Drum
32	OK2
33	BALL
34	Japan Style
35	VOLT
36	AUTO
37	SHOCK
38	ESC
39	Banjo 2
40	Scratching
41	Guitar 2
42	Organ 2
43	Piano 2
44	PASS
45	UP2
46	Record
47	Synth Tom-Toms
48	Cow Bell 2
49	Metro
50	Tri
51	Conga
52	Dance BD
53	Dance SD
54	Dance HH
55	Hit
56	Timpani
57	Chinese Cymbal
58	Mini Cymbal
59	Shaker
60	Bell
61	Japanese Drum
62	Synthesizer
63	Canorus
64	Puff!
65	Nohkan

66	Humandr1
67	Humandr2
68	Dog
69	Cat

### 9-3 MML Commands

Music Macro Language is abbreviated to MML. Commands express elements of music such as intervals, note length, pitch, tone, and volume.

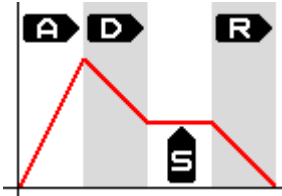
General Music Commands	
: Value	<b>Channel Select(:0-:7)</b> A single piece of music can be played using up to 8 channels. (Please ensure that the channels are written with the lowest number first.)
T Value	<b>Temp(T1-T240)</b>
\$ variable number = value	<b>Replacing Variables</b> Enters a value in place of the designated variable. Variable numbers (0-7), Values (0-255) Use BGMSETV to write from a program, or read using BGMGETV(). These values can be used in commands marked with ●

Commands for Sound Duration	
L Value	<b>Default Sound Duration(L1-L192)</b> To add a dot to a note, follow the desired duration with a full stop e.g. L4.
Q Value	<b>Gate Time (Q0-Q8)</b> This refers to the relationship between the length of time a sound is actually played for and the time it is audible for.
&	<b>Tie(Slur)</b> This refers to joining two sounds together to create a longer sound.

Commands Relating to Pitch and Scale	
CDEFGAB	<b>Designating Scale</b> C=Do, D=Re, E=Mj, F=Fa, G=So, A=La, B=Ti.

C# D# E# F# G# A# B# (# can also be represented by +) C- D- E- F- G- A- B-	For semitones, # or - are added. Notes will be played for the default duration, but by appending lengths directly after the note name, e.g. G4E#16, you can adjust the length of time notes are played for.
R	<b>Rests</b> As with scales, the duration of rests can be appended to the end.
N Value (O4C=40)	<b>Designating Interval Values (N0-N127)</b> Press the button to adjust the value that corresponds to the interval duration.
O Value	<b>Octave (O0-O8)</b> This value changes the octave in which notes are played.
<	<b>Go to Higher Octave</b>
>	<b>Go to Lower Octave</b>
@D Value	<b>Detune (@D-128-@D127)</b> This subtly distorts the sound waves to give a detuned effect.
_(underscore)	<b>Portamento</b> This allows you to slide smoothly between two sound frequencies.

Commands Relating to Volume and Panning	
V Value	<b>Velocity (V0-V127)</b> This adjusts volume when playing scales.
( Value	<b>Increase Velocity ( (0-(127 )</b> This refers to the relationship between the length of time a sound is actually played for and the time it is audible for.
) Value	<b>Reduce Velocity ( (0-(127 )</b>
P Value	<b>Panpot (P0-P127)</b> This value determines the panning between left and right channels. P64=Center.
@V Value	<b>Channel Volume (@V0-@V127)</b> This adjusts channel volume.

Commands Relating to Tone	
@ Value Values above @144 for tone will make this sound louder than the instrumental track.	<b>Tone ( @0-@255 )</b> This adjusts volume when playing scales. This allows you to change the instrument type. 0-127 GM Compatible Instruments 128-129 Drum Set 144-150 PSG (Programmable Sound Generator) 151 Noise 224-255 User Defined Waveforms
@ E Value Attack Value Decay Value Sustain Value Release  Lower ADSR values slow the speed at which the sound is played, while higher values speed it up.	<b>Define Envelope</b> This function lets you adjust the way the sound is played. The value range for each element is 0-127. (e.g.) ""@E0,99,64,64" 
@ER	<b>Deactivating Envelope</b>

Commands Relating to Special Controls	
[	<b>Start Loop</b> Up to 3 elements can be nested within a loop.
] Value	<b>End Loop (0~255)</b> ● This returns the sound being played to before the start of the preceding Start Loop command. (If this value is zero or undefined, the loop will be endless.)

An error will occur when saving MML commands if loop commands do not contain scale, rest, and N values.

## Commands Relating to Macro Definition

{Label=MML}	<b>Macro Definition</b> Definition regularly-used phrases (labels can be up to 8 characters long)
{Label}	<b>Using Macro Definition</b> Use pre-defined MML phrases.

Commands Relating to Performance	
@MON	<b>Modulation ON</b> This activates modulation previously assigned with @MA, @MP.
@MOF	<b>Modulation OFF</b> This cancels modulation.
@MA Value Depth, Value Range, Value Speed, Value Delay	<b>Tremolo</b> This function adjusts the tremolo effect on notes, rapidly repeating the same note. (e.g.) "@MA0,99,64,64"
@MP Value Depth, Value Range, Value Speed, Value Delay	<b>Vibrato</b> This function varies the degree to which a vibrating effect is applied to notes. (e.g.) "@MP0,99,64,64"

@MA and @MP cannot be used at the same time. These effects can be heard as soon as these settings are applied.

#### 9-4 Standard Instruments (@0~@127)

No.	Instrument
0	Acoustic Grand Piano
1	Bright Acoustic Piano
2	Electric Grand Piano
3	Honky-tonk Piano
4	Electric Piano 1
5	Electric Piano 2
6	Harpsichord
7	Clavi
8	Celesta



9	Glockenspiel
10	Music Box
11	Vibraphone
12	Marimba
13	Xylophone
14	Tubular Bells
15	Dulcimer
16	Drawbar Organ
17	Percussive Organ
18	Rock Organ
19	Church Organ
20	Reed Organ
21	Accordion
22	Harmonica
23	Tango Accordion
24	Acoustic Guitar(nylon)
25	Acoustic Guitar(steel)
26	Electric Guitar(jazz)
27	Electric Guitar(clean)
28	Electric Guitar(muted)
29	Overdriven Guitar
30	Distortion Guitar
31	Guitar Harmonics
32	Acoustic Bass
33	Electric Bass(finger)
34	Electric Bass(pick)
35	Fretless Bass
36	Slap Bass 1
37	Slap Bass 2
38	Synth Bass 1
39	Synth Bass 2
40	Violin
41	Viola
42	Cello

43	Contrabass
44	Tremolo Strings
45	Pizzicato Strings
46	Orchestral Harp
47	Timpani
48	String Ensembles 1
49	String Ensembles 2
50	Synth Strings 1
51	Synth Strings 2
52	Voice Aahs
53	Voice Oohs
54	Synth Voice
55	Orchestra Hit
56	Trumpet
57	Trombone
58	Tuba
59	Muted Trumpet
60	French Horn
61	Brass Section
62	Synth Brass 1
63	Synth Brass 2
64	Soprano Sax
65	Alto Sax
66	Tenor Sax
67	Baritone Sax
68	Oboe
69	English Horn
70	Bassoon
71	Clarinet
72	Piccolo
73	Flute
74	Recorder
75	Pan Flute
76	Blown Bottle

77	Shakuhachi
78	Whistle
79	Ocarina
80	Lead 1(square)
81	Lead 2(sawtooth)
82	Lead 3(calliope)
83	Lead 4(chiff)
84	Lead 5(charang)
85	Lead 6(voice)
86	Lead 7(fifths)
87	Lead 8(bass+lead)
88	Pad 1(new age)
89	Pad 2(warm)
90	Pad 3(polysynth)
91	Pad 4(choir)
92	Pad 5(bowed)
93	Pad 6(metallic)
94	Pad 7(halo)
95	Pad 8(sweep)
96	FX 1(rain)
97	FX 2(soundtrack)
98	FX 3(crystal)
99	FX 4(atmosphere)
100	FX 5(brightness)
101	FX 6(goblins)
102	FX 7(echoes)
103	FX 8(sci-fi)
104	Sitar
105	Banjo
106	Shamisen
107	Koto
108	Kalimba
109	Bag pipe
110	Fiddle

111	Shanai
112	Tinkle Bell
113	Agogo
114	Steel Drums
115	Woodblock
116	Taiko Drum
117	Melodic Tom
118	Synth Drum
119	Reverse Cymbal
120	Guitar Fret Noise
121	Breath Noise
122	Seashore
123	Bird Tweet
124	Telephone Ring
125	Helicopter
126	Applause
127	Gunshot

#### 9-5 Drums (@128-@129)

Select tone with @128 (or @129) and percussion will be played to fit with scales between O3B~O5A.

No.	Instrument	
O1B	Acoustic Bass Drum 2	909BD
O2C	Acoustic Bass Drum 1	808BD
O2C#	Side Stick	←
O2D	Acoustic Snare	808SD
O2D#	Hand Clap	←
O2E	Electric Snare	909SD
O2F	Low Floor Tom	808Tom LF
O2F#	Closed Hi-hat	808CHH
O2G	High Floor Tom	808Tom HF
O2G#	Pedal Hi-hat	808CHH
O2A	Low Tom	808Tom L
O2A#	Open Hi-hat	808OHH

O2B	Low-Mid Tom	808Tom LM
O3C	High Mid Tom	808Tom HM
O3C#	Crash Cymbal 1	808Cymbal
O3D	High Tom	808Tom H
O3D#	Ride Cymbal 1	←
O3E	Chinese Cymbal	←
O3F	Ride Bell	←
O3F#	Tambourine	←
O3G	Splash Cymbal	←
O3G#	Cowbell	808Cowbell
O3A	Crash Cymbal 2	←
O3A#	Vibra-slap	←
O3B	Ride Cymbal 2	←
O4C	High Bongo	←
O4C#	Low Bongo	←
O4D	Mute Hi Conga	808Conga MH
O4D#	Open Hi Conga	808Conga OH
O4E	Low Conga	808Conga L
O4F	High Timbale	←
O4F#	Low Timbale	←
O4G	High Agogo	←
O4G#	Low Agogo	←
O4A	Cabasa	←
O4A#	Maracas	808Maracas
O4B	Short Whistle	←
O5C	Long Whistle	←
O5C#	Short Guiro	←
O5D	Long Guiro	←
O5D#	Claves	808Claves
O5E	Hi Wood Block	←
O5F	Low Wood Block	←
O5F#	Mute Cuica	←
O5G	Open Cuica	←
O5G#	Mute Triangle	←

O5A	Open Triangle	←
-----	---------------	---

#### 9-6 PSG and Waveforms (@144-@255)

No.	Instrument
144	PSG - Duty Rate 12.5%
145	PSG - Duty Rate 25.0%
146	PSG - Duty Rate 37.5%
147	PSG - Duty Rate 50.0%
148	PSG - Duty Rate 62.5%
149	PSG - Duty Rate 75.0%
150	PSG - Duty Rate 87.5%
151	Noise
224 : 255	*Waveforms created by the user with the BGMPRG command

## 10 BASIC Standard Features

Petit Computer Standard features and Limitations.

### 10-1 Basic Components

Characters	Multi-byte characters are used.
Character Types	Numbers, alphabet and symbols are available.
Numerical Values	32 bit fixed-point numbers are used (with fractions rounded up). 4096 is treated as 1.0 Integers within the range of $\pm 524287$ can be used. Values outside the range will not be recognized.
Hexadecimal (Base 16) Notation	&H
Binary Notation	&B
Variables	Should be expressed within a maximum of 16 characters, starting with alphabetical characters. The character _ will also be recognized. For string variables, add \$ at the end of the name. (ex.) ANS=75:C\$="TEXT"
Array	There is a total of 32768 possible elements, in two dimensions. Parentheses are either () or []. Subscript starts from 0. (ex.) The range of DIM NO(10) is from NO(0) to NO(9) <b>A maximum of 4096 variables can be used in character strings.</b> <b>The defined array and the number of variables actually used may vary.</b> <b>If the maximum is exceeded, it will result in an error.</b>
Multiple Commands	Possible (use : (colons) to split)
Sub-Routines and Nesting	No limits. FOR~NEXT can be used in same way (within limits of available memory)
File Control Structure	When files are being saved or loaded, users will be unable to input commands while the dialog box is displayed.
Save/Load Units	Resource units (The user cannot freely load and save files as they require MEM resources).

File Names	Must be a maximum of 8 characters, starting with a letter of the alphabet. ( 'A'~'Z', '_', '0'~'9' )
------------	---

## 10-2 Operator symbol (Arithmetic-Compare-Bit)

There are a total of 5 arithmetic operators:

+	Addition (A+B)
-	Subtraction (A-B)
*	Multiplication (A*B)
/	Division (A/B) *Division by 0 produces error
%	Remainder (A%B) *Division by 0 produces error

Addition and multiplication can be used in character string variables.

(e.g.) String for Addition

```
A$=""ABC"" : B$=""XYZ"" : PRINT A$+B$
ABCXYZ
```

(e.g.) String for Multiplication

```
A$=""ABC"" * 4: PRINT A$
ABCABCABCABC"
```

There are a total of 6 relational operators:

>	Value on left is greater than value on right (A>B)
<	Value on left is less than value on right (A<B)
>=	Value on left is equal or greater than value on right (A>=B) NOTE => will not function
<=	Value on left is equal or less than value on right (A<=B) NOTE =< will not function
==	The values are equal (A==B)
/=	The values are not equal (A!=B)

There are a total of 4 bitwise operators:

AND	Logical AND (A AND B)
OR	Logical OR (A OR B)
XOR	Exclusive OR (A XOR B)
NOT	Negation (NOT A)



This operator inverts what is true and false.

!	True/False Inversion Symbol * !TRUE is same as FALSE * !FALSE is same as TRUE
---	---

#### Order of Operations

1	Sections inside ( ) [ ]
2	Minus, NOT
3	Functions
4	* / % (multiplication, division, remainders)
5	+ -
6	== != < <= > >=
7	AND, OR, XOR

#### 10-3 Edit Functions

The text memory can handle approximately 520,000 characters. If the total amount of characters exceeds this, it will not be possible to save programs, even if they are within the maximum number of lines.

Editor	The text editor in Edit Mode allows you to edit one line at a time. Pressing ENTER will automatically assign a line number. Text will not be wrapped.
Line Limitations	1-9999 (when individual lines are very long, you may be unable to use the maximum number of lines).
Line Numbers	Line numbers are treated as lines in a text editor. When new line numbers are assigned automatically when a linebreak is added, commands like GOTO and GOSUB will not be able to designate specific line numbers. For branch instructions, the command @LABEL NAME is used instead of line numbers.
Characters Per Line	Up to 100 characters can be entered on a single line. The line will scroll horizontally to display characters that cannot be displayed on screen. (If there is insufficient text memory, you may not be able to enter the maximum 100 characters on a single line).

#### 10-4 Entry Methods

Keyboard	Software keyboard (alphanumeric characters and symbols)
----------	--

Hardware Buttons	Can be used. (SELECT is used as the ESC key. The L Button and START can only be used when a program is running).
Touch Panel	Can be used, utilizing TCHST, TCHX, TCHY as system variables.

#### 10-5 Display

Display Order Priority	The user's sprites can be displayed in front of the user's BG screens.							
Upper Screen Display Levels	<div>Front</div> Console Screen User BG Screen (Front) User Sprite Screen <div>Back</div> User BG Screen (Back) Graphic Screen Backdrop Screen							
Lower Screen Display Levels	<div>Front</div> Keyboard or Panel User BG Screen (Front) User Sprite Screen <div>Back</div> User BG Screen (Back) Graphic Screen Backdrop Screen							
Graphic Resolution	256x192 pixels							
Characters on Console Screen	32 chars. x 24 lines (if linebreak is added to last line, it will scroll an additional 1 line)							
Animation Functions	Simple animations are possible using sprite commands.							
Character Functions	Images consisting of 8 x 8 pixel units. They are SPRITE and BG resources (CHR style). <table><tr><td>BGU</td><td>User Bank x 4 (BGU0-BGU3)</td></tr><tr><td>SPU</td><td>User Bank x 8 (SPU0-SPU7)</td></tr></table>		BGU	User Bank x 4 (BGU0-BGU3)	SPU	User Bank x 8 (SPU0-SPU7)		
BGU	User Bank x 4 (BGU0-BGU3)							
SPU	User Bank x 8 (SPU0-SPU7)							
Number of Colors Displayed Simultaneously	Colors can be assigned independently to the upper and lower screens. The total number of colors is displayed below: <table><tr><td>BG</td><td>16 colors x 16 types (including transparent)</td></tr><tr><td>SP</td><td>16 colors x 16 types (including transparent)</td></tr><tr><td>GRP</td><td>256 colors (0 is transparent)</td></tr></table>		BG	16 colors x 16 types (including transparent)	SP	16 colors x 16 types (including transparent)	GRP	256 colors (0 is transparent)
BG	16 colors x 16 types (including transparent)							
SP	16 colors x 16 types (including transparent)							
GRP	256 colors (0 is transparent)							

Palette	Colors for SPRITE and BG are divided into units called palettes. Palettes each have 16 colors, including a transparent one, each of which has a number assigned to it.
Text Color	The 15th color in the designated palette for SPRITE and BG is used for text color. If this color is changed, the color of the text on screen will change accordingly.
Background Color	The color numbered 0 taken from the BG palette numbered 0 is used as the background color.

#### 10-6 Audio Function

Number of Simultaneous Channels	There are a total of 16 channels available for music, sound effects and synthesized voices (not including PSG).
Sound Effects	There are 70 preset sound effects. Volume, panning and frequency can be modified. 8 sounds can be played at the same time.
Voice Synthesis	Text entered is read using voice synthesis. The emotion, pitch, and speaker type can all be modified.
Background Music Source	128 types of instrument 2 types of drum kit PSG and sound sources User-defined simple waveform sources
Background Music Waveform Definition	Adapts to one of 32 defined waveforms.
BGM	30 Preset Songs 128 User-Defined Songs Maximum of 8 Songs Played Simultaneously

#### 10-7 Error Number Chart

There are a range of different errors and warnings which can occur when a BASIC program is run. When an error occurs, the error number is given after the system variable ERR. The line number is given in ERL messages.

1	Syntax error	There is problematic grammar in the program.
2	Out of range	The value exceeds the valid range.

3	Out of memory	There is insufficient memory available.
4	Undefined label	The destination for a branch instruction cannot be located.
5	NEXT without FOR	There is a NEXT command which does not belong to any FOR command.
6	Out of DATA	There is insufficient DATA available for a READ command.
7	Illegal function call	There is a problem with the assignment of elements in a function or command.
8	Duplicate definition	The same array or variable has been defined more than once.
9	Can't continue	A program cannot be continued using a CONT command.
10	Missing operand	There are insufficient parameters.
11	Duplicate label	The same label has been defined more than once.
12	Illegal resource type	The resource type designated by a string does not exist.
13	Illegal character type	The designated character type does not exist.
14	String too long	The string is too long. Labels should be no longer than 8 characters, while strings should be no more than 256 characters in length.
15	Division by zero	A number has been divided by zero.
16	Overflow	The results of an operation have exceeded the permitted range.
17	Subscript out of range	The subscript for an array variable is out of range.
18	Type mismatch	Variable types do not match.
19	Formula too complex	The formula may have too many bracketed sections, or otherwise be too complex.
20	RETURN without GOSUB	A RETURN command is present without an accompanying GOSUB command.
21	FOR without NEXT	A FOR commands is present which does not correspond to a NEXT command.
22	Illegal MML	There is an error in the MML.

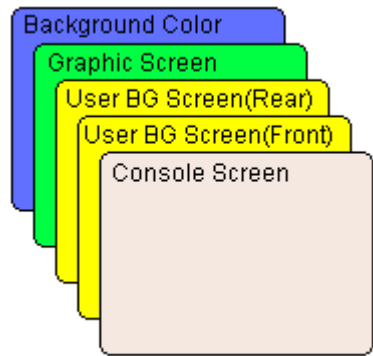
11 Text and Display

Information on the keyboard, text, as well as screen display and colors.

11-1 Display Screen Structure

Upper Screen

The upper screen is composed of five separate layers, listed from the back: the background color screen, the graphic screen, the user's rear BG (background) screen, the user's foremost BG (background) screen, and the console screen.



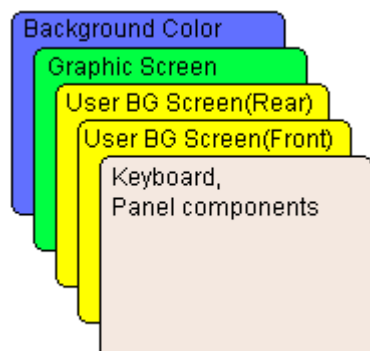
While user's programs are running, SPSET or SPCHR commands can be used to adjust the order of precedence in which sprites are displayed on the graphic, front and rear BG screens.

Background Color	The monochrome screen displayed behind every screen.
Graphic Screen	The screen which you can color or draw lines, curves and more on.
User BG Screen (Rear)	The BG screen displayed at the rear (8x8 characters, in 64x64)
User BG Screen (Front)	The foremost BG screen (8x8 characters, in 64x64)
Console Screen	This screen is where the keyboard, control panel, file select etc. are displayed

Lower Screen

The lower screen is composed of five separate layers, listed from the back: the background color screen, the graphic screen, the user's rear BG (background) screen, the user's foremost BG (background) screen, the keyboard, and the panel

components. It normally displays the keyboard screen, but when running programs, background and graphics can be used via PNLTYPE commands.



Background Color	The monochrome screen displayed behind every screen.
Graphic Screen	The screen on which you can color or draw lines, curves and more.
User BG Screens (Front - Rear)	User BG Screens
Keyboard, Panel components	This screen is where the keyboard, control panel, file select etc. are displayed

## 11-2 Color Palette

In this software, the console, background screens, and sprites and graphics are all distinct graphical elements:

- Console characters and BG (background) screens
- Sprites
- Graphics

For each element, a separate color palette is available. For the BG screens, characters and sprites, colors are divided into units of 16 (the color code 0 is transparent). For graphics, a palette of 256 colors can be used.

Each color available for the BG screens and sprites is divided into 16 with color codes along the top, and palette codes along the side.

Use the COLSET command to change the palette to your preferred colors. In general, if you keep darker shades on the left side of the screen, and have brighter colors as you move towards the right, this will avoid having a confusing mix of colors in your palette."

#### ◆ For Console Characters, Background or Sprites

Palette ↓ / Color code→

(Color code 0 = Transparent)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

Colors are divided, with 3 colors assigned for each application (App. 1-4). Colors are defined as dark, normal and bright.

#### ◆ Graphics

Palette ↓ / Color code→

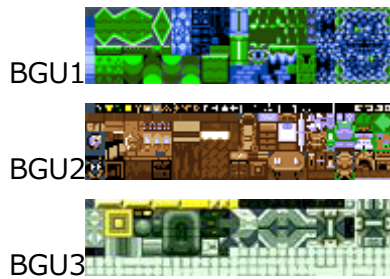
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

### 11-3 User BG Characters

Select parameters with LOAD and use this as the background for your screen. You can restore the initial settings using CHRINIT.



BGU0




### 11-3 User Sprite Characters

When you wish to display a character on the sprite screen and have it move in four directions, select each of the four directions (up, down, left, right) by assigning a sprite character number from the SPU resources.



For example, for SPU1 resources, inputting sprite character numbers 64-67 (right), 68-71 (down), 72-75 (left) and 76-79 (up) will cause the sprite to move in a clockwise direction on the screen. The SPANIM command allows simple animation. Character images can be freely adjusted by users.

### 11-5 Keyboard Display

Touch a string and that string will be displayed on the console screen. Touch SHIFT or press  to change the position of the string.

#### Alphabet



#### Alphabet + SHIFT (or )





## Symbols



## Symbols + SHIFT (or )



## Kana



## Kana + SHIFT (or )



## 11-6 Character Code List

The following 256 characters are available.



Unlike previous BASIC programs, this program does not contain control codes from 0-31.

## **12 Files and Resources**

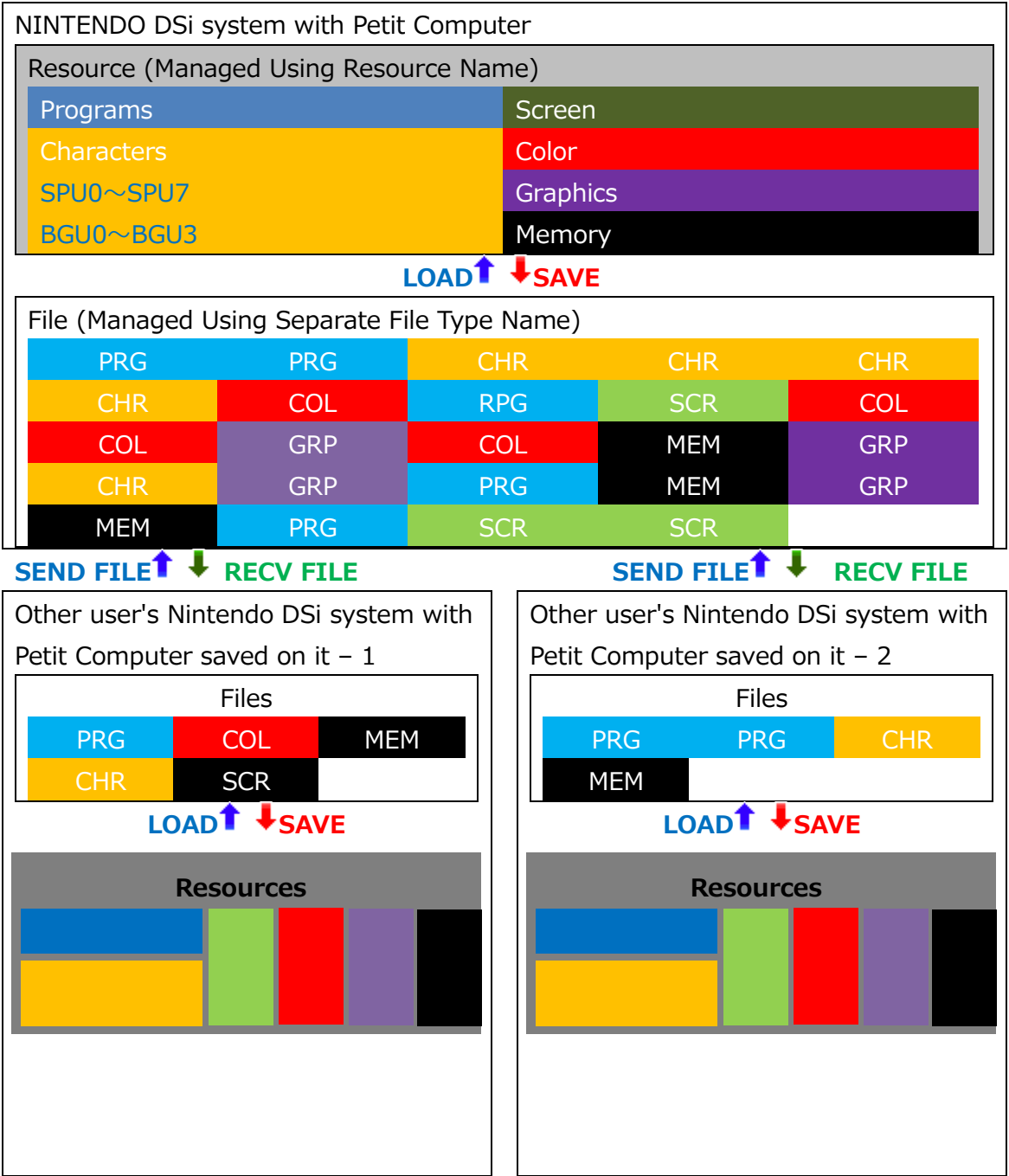
This section gives more information about accessing files within programs, sending files saved using the Petit Computer to other DSi systems, and receiving files from other users.

Commands in this section:

**LOAD, SAVE, SENDFILE, RECVFILE**

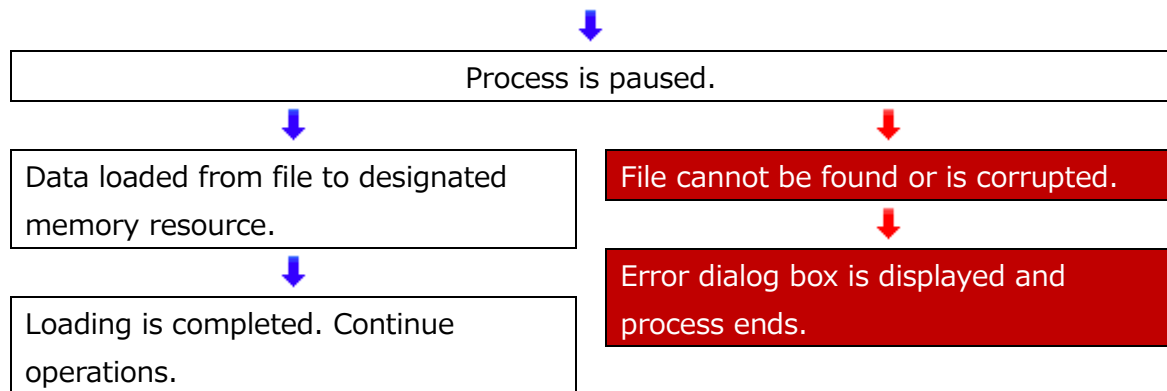
12-1 The Relation Between Files and Resources

LOAD and SAVE commands read and write between internal resources and files, while RECVFILE and SENDFILE commands allow files to be transferred between DSi systems which have the Petit Computer software.

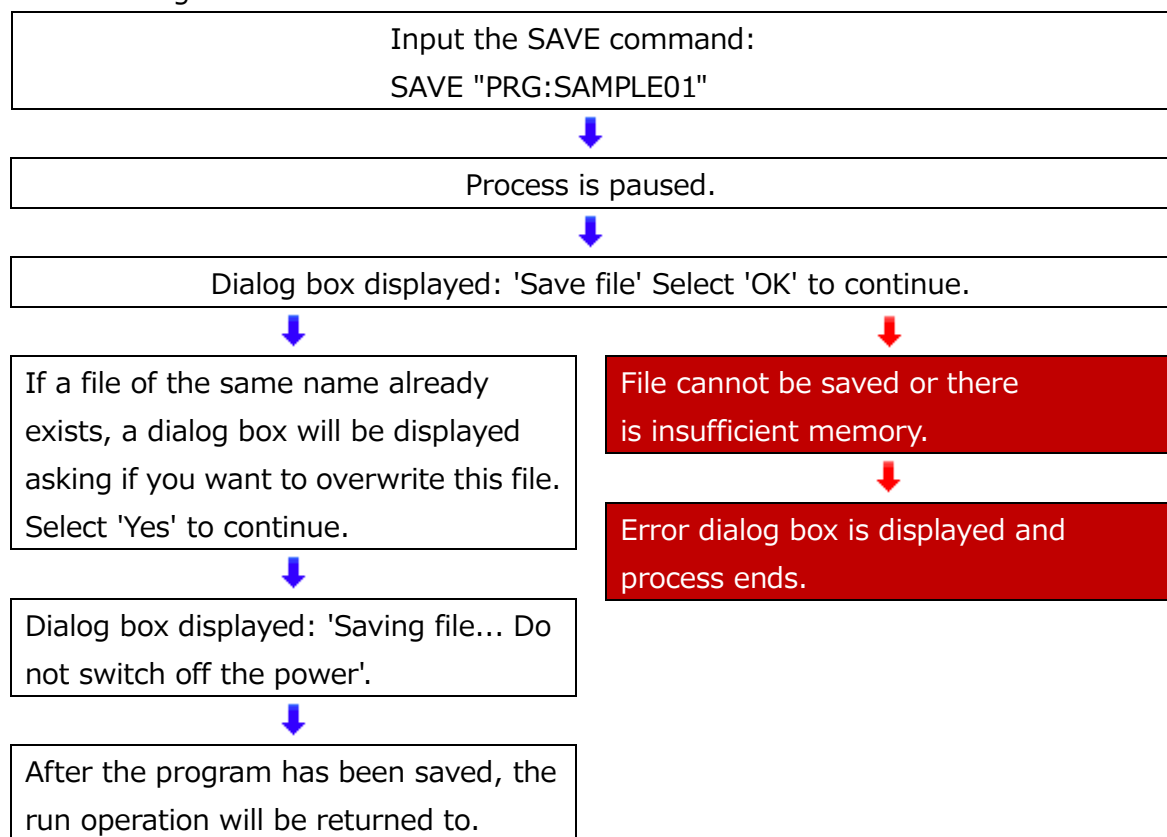


12-2 Loading

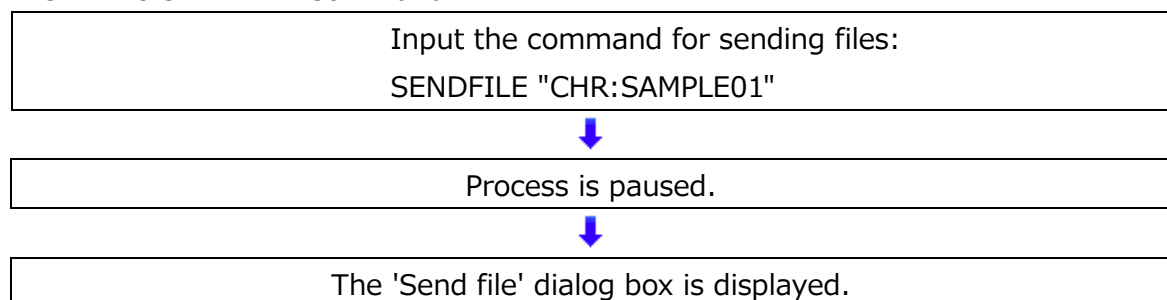
Input the LOAD command:  
LOAD "PRG:SAMPLE01"

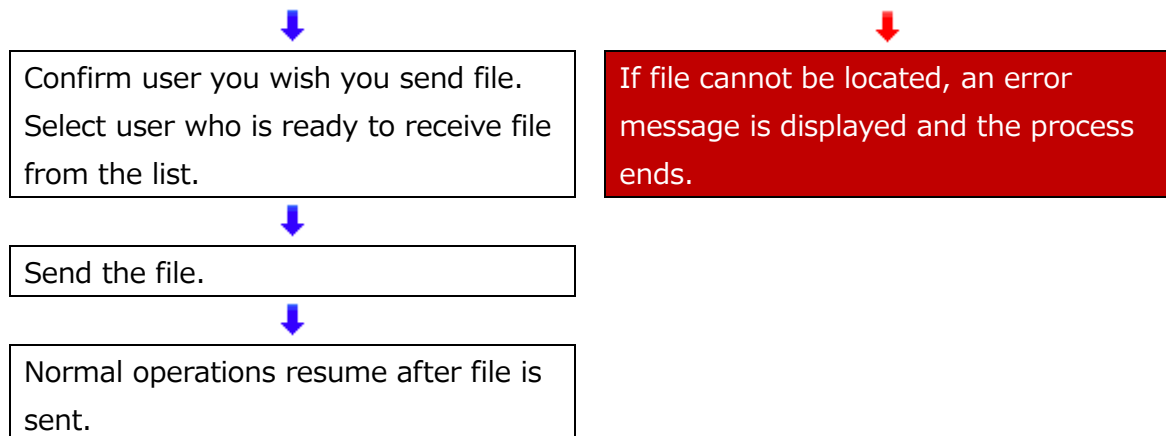


### 12-3 Saving

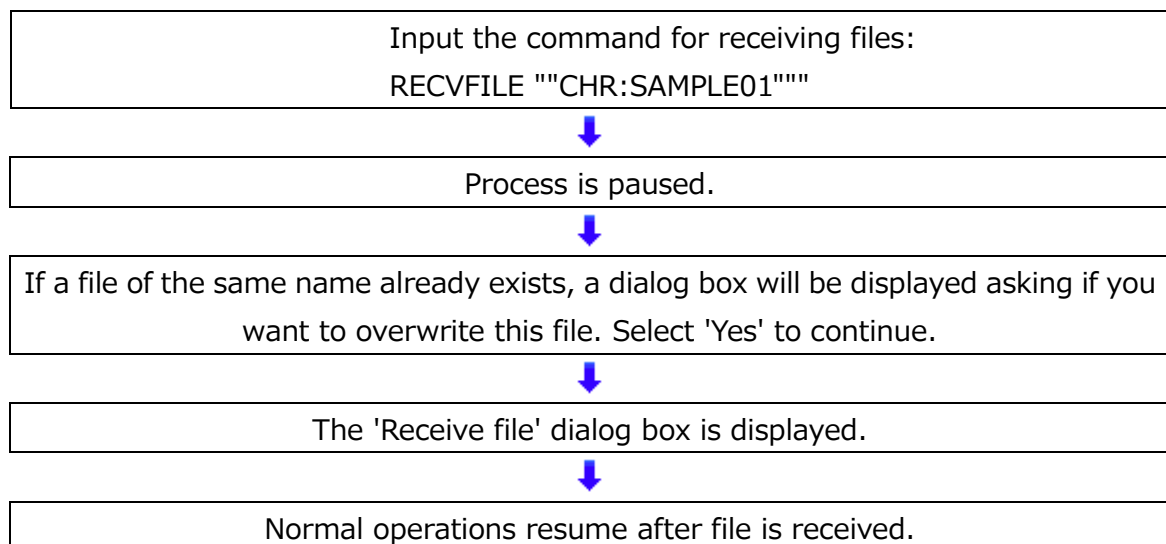


### 13-4 The SENDFILE Command





#### 12-5 The RECVFILE Command



## 13 Rules for Commands and Functions

This a list of the rules concerning the commands and functions included in BASIC.

### 13-1 Petit Computer Commands

In order to give instructions to the computer, a uniform style for writing commands has been used.

#### ©Command Format

Command [ parameter ] [, parameter ] [, parameter ]...

### 13-2 Specific Style for Commands

Petit Computer has used its own original style of writing commands, rather than simply following the style of commands from previous BASIC programs.

#### ©Previous BASIC style (prioritizing visibility)

LINE( x,y )-( x2,y2 ),color

#### ©Original Petit Computer style

GLINE x,y,x2,y2,color

### 13-3 Petit Computer Functions

In Petit Computer, commands can be used without parentheses ' ( '. This means that functions directly after a command should use parentheses ' ( ' .

#### ©Function

Variable = COLGET\$( ""Type Name"", Number)

### 13-4 Commands with Multiple Values

#### Intermediate position between functions and commands

In the C programming language, functions which return multiple parameters can use pointers to pass the data. However, this is not an option in BASIC. This means that each piece of returned data requires its own function. In this software, a READ function has been implemented that allows multiple pieces of information to be carried over.

#### ©Command obtaining multiple values

TMREAD( "<time string>" ) ,H ,M ,S

When obtaining numbers, it is often necessary to have parameters. This is why Petit Computer also allows parameters to be passed to a function. Petit Computer was not designed to accommodate the parameter type or the number of parameters changing.

H

Variable that retrieves hour

M

Variable that retrieves minute

S

Variable that retrieves seconds

## 14 Simple Alphabetical Listings

The commands and system functions in SmileBASIC are grouped into their respective abbreviations, and are listed here in alphabetical order. For basic command names, typing a single letter into the keyboard will bring up a list of suggestions. Consult this table to see more details of the number and contents of parameters.

A
Variable=ABS (number)
ACLS
AND
APPEND "PRG: file name"
Variable=ASC (character)
Variable=ATAN (radian value)
B
BEEP [ waveform number [,pitch [,volume [,panpot]]]]
Variable=BGCHK()
BGCLIP start x, start y, end x, end y
BGCLR [Layer]
BGCOPY layer, start x, start y, end x, end y, destination x, destination y
BGFILL layer, start x, start y, end x, end y, character number, palette number, horizontal rotation, vertical rotation <a href="#">BGFILL layer, start point x, start point y, end x, end point y, screen data</a> BGFILL layer, start x, start point y, end point x, end point y, "screen data string"
Variable=BGMCHK( [track number] )
BGMCLEAR [song number]
Variable=BGMGETV (track number, variable number)
BGMPLAY song number <a href="#">BGMPLAY track number, song number, [,track volume]</a> BGMPLAY "MML string"[,"MML string"...]
BGMPRG waveform number,A,D,S,R,"waveform string"
BGMSET song number,"MML string"[,"MML string"]
BGMSETD song number,@label <a href="#">BGMSETD song number,variable\$</a>



BGMSETV track number,variable number,value
BGMSTOP [ track number] [, fade time]
BGMVOL [track number,] volume
BGOFS layer,x,y [,interpolation time]
BGPAGE screen
BGPUT layer, x, y, character number, palette number, horizontal rotation, vertical rotation BGPUT layer, x, y, screen data BGPUT layer, x, y,"screen data string"
BGREAD (layer,x,y), CHR, PAL, H, V BGREAD (layer, x, y), SC BGREAD (layer, x, y),SC\$
BREPEAT button ID [, start time, interval]
Variable=BTRIG()
Variable=BUTTON( [sorted])
C
CANCEL
Variable=CHKCHR( x coordinate,y coordinate )
Variable \$ =CHR\$(character code)
CHRINIT "character name"
CHRREAD ("character name",character number),C\$
CHRSET "character name",character number, "graphic string"
CLEAR
CLS
COLINIT "color bank name", color number
COLOR palette number
COLREAD ("color bank name", color number),R,G,B
COLSET "color bank name",color number,"color data string"
CONT
Variable=COS (radian value)
CSRX
CSRY
D
DATA number, number,... DATA "string", "string"...

DATA 123,345,56,"SAMPLE"
DATE\$
Variable=DEG(radian value)
DELETE "resource name:file name"
DIM pos[4]
<color rgb5=00001f>DIM sample[10,5]
DTHREAD ("date string"),YEAR,MON,DAY
E
ELSE
END
ERL
ERR
EXEC "PRG:file name"
Variable=EXP (number)
F
FALSE
FILES [resource name[,resource name...]]
Variable=FLOOR (number)
FOR variable=initial value TO final value [STEP increase]
FREEMEM
FREEVAR
FUNCNO
G
GBOX start x, start y, end x, end y [,color]
GCIRCLE x, y, radius [,color [, initial angle, final angle]]
GCLS [color]
GCOLOR color number
GCOPY [transfer page,] start x, start y, end x, end y, destination x, destination y, copy mode
GDRAWMD status
GFILL start x, start y, end x, end y [,color]
GLINE start x, start y, end x, end y [,color]
GOSUB @label
GOSUB variable\$

GOTO @label GOTO variable \$
GPAGE screen [, drawing page][, display screen]
GPAINT x, y [, color [, border color]]
GPRIO number
GPSET x, y [,color]
GPUTCHR x, y, "character name", number, palette number, scale
Variable=GSPOIT(x,y)
H
Variable\$=HEX\$ (numerical value[, decimal places])
I
Numerical value=ICONCHK()
ICONCLR [ icon position ]
ICONPAGE
ICONPMAX
ICONPUSE
ICONSET icon position,icon number
When the IF condition is met, the designated command following THEN will be performed. IF condition is met THEN @label When the IF condition is met, the designated command following THEN will be performed. When these conditions are not met, the command following ELSE will be performed. IF condition THEN @label ELSE @label
IF (condition) GOTO @label IF condition is met GOTO @label. If condition is not met, the command after ELSE is performed. IF condition GOTO @label ELSE @label
Variable\$=INKEY\$()
INPUT ["string";] received variable INPUT ["string";] received variable\$ INPUT ["string";] received variable, received variable\$
Variable = INSTR ("string", "search target string")
K

KEY number, "string"
KEYBOARD
L
@label name
Variable\$ = LEFT\$ ("string", character count)
Variable=LEN(string)
LINPUT ["string";] received variable\$
LIST
LIST @label
LIST line number
LOAD "resource name:file name" [,display control]
LOCATE x,y
Variable=LOG(numerical value)
M
MAINCNTL
MAINCNTH
MEM\$
Variable\$=MID\$(string,initial position,character count)
N
NEW
NEXT [variable name]
NOT
O
ON variable GOSUB @label0,@label1,@label2...
ON variable GOTO @label0,@label1,@label2...
OR
P
PACKAGE\$
Variable=PI()
PNLSTR x coordinate,y coordinate, "string" [, palette number]
PNLTYPE "panel name"
Variable = POW (numerical value, exponential value)
PROGRAM\$

PRINT "string"
PRINT variable
PRINT variable\$
PRINT variable;variable\$;"string"
PRINT "string",variable,variable\$
<b>R</b>
Variable=RAD(angle)
READ obtained variable1[,obtained variable2...]
READ A
READ B\$
READ X,Y,Z,G\$
REBOOT
RECVFILE "resource name:file name"
REM The following is a comment 'commenttext
RENAME "resource name:file name","new name"
RESTORE @label
RESTORE variable\$
RESULT
RETURN
Variable\$ = RIGHT\$ ("string", character count)
Variable=RND(max value)
RSORT start point, number of elements, array 1 [, array 2...]
RUN
<b>S</b>
SAVE "resource name:file name"
SENDFILE "resource name:file name"
Variable=SGN(variable)
Variable=SIN(radian value)
SORT start point, number of elements, array 1 [, array 2...]
SPANGLE control number,angle [,interpolation time,change direction]
SPANIM control number,number of frames,time [,loop]
Variable=SPCHK(control number)
SPCHR control number, sprite character number [,palette number, horizontal rotation, vertical rotation, order of precedence]

SPCLR [control number]
SPCOL control number, x, y, w, h, scale adjustment [, group]
SPCOLVEC control number [, displacement x, displacement y]
variable = SPGETV (control number, variable number)
variable = SPHIT (control number [, initially determined control number])
SPHITNO
variable = SPHITRC (control number, x, y, w, h [, displacement x, displacement y])
variable = SPHITSP (control number, control number of other user)
SPHITT
SPHITX
SPHITY
SPHOME control number, x, y
SPOFS control number,x,y [,interpolation time]
SPPAGE screen
SPREAD(control number), X, Y [, A, S, C]
SPSCALE control number,scale [,interpolation time]
SPSET Control number, sprite character number, palette number, horizontal rotation, vertical rotation, order of precedence [, width, height]
SPSETV control number, variable number, value
Variable = SQR(number)
STEP
STOP
Variable\$ = STR\$(number)
Variable\$ = SUBST\$("string", start point, character number, "replacement string")
SWAP variable, variable SWAP variable\$, variable\$
SYSBEEP
T
TABSTEP
Variable=TALKCHK()
TCHST
TCHTIME
TCHX
TCHY

THEN
TIME\$
TMREAD ("time string"),HOUR,MIN,SEC
TO
TRUE
V
Variable=VAL(string)
VERSION
VISIBLE console,panel,BG0,BG1,SPRITE,256 color graphics
VSYNC frame number
W
WAIT frame number
X
XOR

## 15 System Variables

### 15-1 Numeric System Variables

	R	W	(R=Read, W=Write)
CSRX	x		Current cursor x-axis (horizontal) position
CSRY	x		Current cursor y-axis (vertical) position
FREEMEM	x		Remaining memory available to user
VERSION	x		System Version (0xAABBCCDD, Version AA.BB.CC.DD)
ERR	x		Error number immediately after error occurred
ERL	x		Number of line where error occurred.
RESULT	x		Result of running file-related command
TCHX	x		x coordinate pressed on Touch Screen.
TCHY	x		y coordinate pressed on Touch Screen.
TCHST	x		Touch status (TRUE = Touched)
TCHTIME	x		Time Touch Screen is touched for (Given in number of frames)
MAINCNTL	x		Frames elapsed since start of program (max. 145 minutes)
MAINCNTH	x		Duration of frame display since program launched (data over 145 minutes)
TABSTEP	x	x	Amount TAB key will move(0-16)
TRUE	x		Always 1
FALES	x		Always 0
CANCEL	x		Always -1
ICONPUSE	x	x	FALSE=Don't use TRUE=Use
ICONPAGE	x	x	Page number for user system icon (0 is always entered in Run Mode)
ICONPMAX	x	x	Maximum number of pages for user system icon (Does not work in Run Mode.)
FUNCNO	x		Number of function key pressed (1-5, 0=not pressed)
FREEVAR	x		Number of variables that can be saved
SYSBEEP	x	x	System sound effect controls (True = ON, False =



			OFF)
KEYBOARD	x		Key Scan Code
SPHITNO	x		SPRITE Collision Detection Results (-1=None, 0-99=Detection)
SPHITX	x		Sprite Detection x Coordinate
SPHITY	x		Sprite Detection y Coordinate
SPHITT	x		Sprite Collision Time

### 15-2 Text String System Variables

R W (R=Read, W=Write)			
TIME\$	x		Obtains current time as a string (HH:MM:SS)
DATE\$	x		Obtains current date as a string (YYYY/MM/DD)
MEM\$	X	x	Number of strings that can be saved in file
PRGNAME\$	x		The LOAD, EXEC, RECVFILE parameters run most recently will be stored.
PACKAGE\$	x		The package data for the last file read by the system is stored.

### 15-3 KEY Scan Code



The system variable KEYBOARD is a special variable that allows the gathering of keyboard data that cannot be gained from INKEY\$. The values gained are not numbers corresponding to characters, but are values that correspond to keys on the keyboard. If nothing is inputted, the KEYBOARD variable will be 0.

## 16 Run Mode-Specific Commands

There are 5 commands that can only be used in Run Mode. These are related to running and continuing programs and cannot be written into the programs themselves:

**NEW, LIST, RUN, CONT, FILES, REBOOT**

### 16-1 NEW

This deletes the program.

Programs you are currently editing will be lost. If you wish to use them again, store them using a **SAVE** command before exiting.

Format	NEW	
Parameters	None	
Returns	None	
Error		

### 16-2 LIST

Switches to Edit Mode and begins edit.

Format	LIST	
	LIST @label	
	LIST line number	
Parameters	Line number	Designate line where source is displayed.
	@label	Designate line where source is displayed.
Returns	None	
Error	When line number or label does not exist.	

### 16-3 RUN

This runs the program.

Format	RUN	
Parameters	None	
Returns	None	
Error		

## 17 CONT

Continues a program stopped with a **STOP** command.

Format	CONT
Parameters	None
Returns	None
Error	When program has been run and cannot be continued.

#### 16-5 FILES

Format	FILES [file type name [, file type name...]]	
Parameters	File Type Name	
	Strings are allocated according to the resource type of the target file.	
	PRG	Program (can be omitted)
	MEM	Memory
	COL	Color
	GRP	Graphics
	SCR	User screen
	CHR	User character
Returns	None	
Error		

#### 16-6 REBOOT

Exit BASIC and return to the Home Menu.

Programs you are currently editing will be lost, as well as other data, including characters you are creating and color settings. If you wish to use any elements from your current program again, please use the SAVE command.

Format	REBOOT
Parameters	None
Returns	None
Error	

## 17 Declarations & Assignment Commands

The following commands allow you to initialize memory, perform array declarations and more:

CLEAR, =, DIM, REM, @, KEY

### 17-1 CLEAR

This resets variable names and BASIC internal memory.

Format	CLEAR
Parameters	None
Returns	None
Error	

### 17-2 =(LET)

Assign (an abbreviation of the LET command)

Format	ABC=123 TEXT\$="ABCDE"
Parameters	None
Returns	None
Error	

### 17-3 DIM

Array declarations

Element count for up to 2 dimensions.

Element count can be defined up to a total of 262144."

Format	DIM pos(4),size(4) DIM sample(10,5) DIM MSG\$(15)
Parameters	None
Returns	None
Error	

### 17-4 `(REM)

For annotations(comments).

The text following this command up to the next linebreak will be ignored.

Format	REM The following is a comment ` comment text
Parameters	None

Returns	None
Error	

#### 17-5 @(Label definition)

Declaration of the name, giving the line number.

This always needs to be added at the start of a row. The character string following @ will become the name under which it is saved. It can be used to give destinations with commands such as GOTO and GOSUB.

Format	@label name	
Parameters	Label name	Unlike strings, it is not necessary to enclose with ""
Returns	None	
Error	When characters or symbols that are not alphanumeric characters or '_' are in a label name. When the label name is not defined.	

#### 17-6 KEY

Assigning Strings to Function Keys

When used in a user-created program, the string data assigned to the function key will be entered in the program when that key is pressed.

Format	KEY number, "string"	
Parameters	Number	Function Key Number (1-5)
	String	Only 4 characters from the assigned string will be displayed, but strings of up to 256 characters can be saved. When the full string cannot be displayed, the last character will be displayed as '.'
Returns	None	
Error	When a number is designated that does not exist.	

## 18 Variable Controls & WAIT Command

The following basic commands can be used to control variables:

SWAP, SORT, RSORT, VSYNC, WAIT

### 18-1 SWAP

This swaps the contents of 2 variables.

Format	SWAP variable 1, variable 2 SWAP variable \$1, variable \$2	
Parameters	Variable 1	First variable
	Variable 2	Second variable
Returns	None	
Error		

### 18-2 SORT

This arranges values in ascending order (1→99) in a one-dimensional array.

Format	SORT start position, number of elements, array 1 [ , array 2...]	
Parameters	Start Position	Start Point to Arrange Values (0 ~
	Number of Elements	Total to be Arranged
	Array 1	Name of Target Array *( ) is not required.
	[ array 2 ]...	Array results will be in order, starting with array 1 (when array names are entered, all arrays will be arranged according to the array 1 results).
Returns	None	
Error	This routine with arrange the contents of array IX in ascending order according to the contents of array V:  CLEAR DIM IX(10),V(10) FOR I=0 TO 9 IX(I)=I : V(I)=RND(100 ) PRINT I;"=";IX(I);";";V(I)	

	<pre> NEXT SORT 0,10, V, IX PRINT FOR I=0 TO 9   PRINT I;"=";IX(I);":";V(I) NEXT </pre>
--	---

### 18-3 RSORT

This arranges values in descending order (99→1) in a one-dimensional array.

Format	RSORT start position, number of elements, array 1 [ , array 2 ... ]	
Parameters	Refer to SORT	
Returns	None	
Error		

### 18-4 VSYNC

Same length as screen renewal time (waiting for graphics to be renewed).

Format	VSYNC Frame number	
Parameters	Frame number	Indicates number of frames since the VSYNC command immediately beforehand. (0 = ignore)
Returns	None	
Error		

### 18-5 WAIT

A simple wait command.

Format	WAIT frame number	
Parameters	Frame number	The program will wait for the designated number of frames. (Setting the frame count to 60 will result in the program waiting for 1 second).
Returns	None	
Error		

## 19 Branch Instructions

The following commands allow you to call routines and set branch instruction conditions:

ON~GOTO, ON~GOSUB, GOTO, GOSUB, RETURN, STOP, END

### 19-1 ON~ GOTO

Causes program to branch depending on numeric values.

Format	Line number when ON variable GOTO variable =0 (or @label), numeric value 1, numeric value 2...	
Parameters	Variables	Branch Number (0~
Returns	None	
Error		

### 19-2 ON~ GOSUB

Calls a sub-routine based on number

Format	Line number (or @label) when ON variable GOSUB variable =0, numeric value 1, numeric value 2...	
Parameters	Variables	Branch Number (0~
Returns	None	
Error		

### 19-3 GOTO

Forced branch.

Instead of an @label, you can also use a text string variable substituted for the label name.

Format	GOTO @label GOTO variable \$	
Parameters	@label	Branch Destination Name
Returns	None	
Error		

### 19-4 GOSUB

Call sub-routine

Instead of an @label, you can also use a text string variable substituted for the label name.

Format	GOSUB @label
--------	--------------



	GOSUB variable \$	
Parameters	@label	Branch Destination Name
Returns	None	
Error		

#### 19-5 RETURN

Returns from a sub-routine.

Always use this command in conjunction with GOSUB.

Format	RETURN
Parameters	None
Returns	None
Error	

#### 19-6 STOP

Forces the currently running program to stop and returns to the console.

Format	STOP
Parameters	None
Returns	None
Error	

#### 19-7 END

Ends the program.

Format	END
Parameters	None
Returns	None
Error	

## 20 Repeat/Comparison Commands

The following commands allow you to repeat routines the designated number of times or judge conditions:

FOR~TO~STEP, NEXT,  
IF~THEN, IF~GOTO

### 20-1 FOR~ TO~ STEP

Repeat the designated number of times

If STEP has been omitted, it will be treated as STEP1. If increase is added and the final value is less than the initial value, the FOR command will be skipped and the NEXT and subsequent commands will be run.

Format	FOR variable = initial value TO final value [STEP increase]	
Parameters	Variables	For Frequency Management
	Initial Value	Number at Start
	Final Value	Number at End
	Increase	Value to Add Once
Returns	None	
Error		

### 20-2 NEXT

End of Loop

Always use as in conjunction with a FOR command

Format	NEXT [variable name]	
Parameters	Variable name	Repeated variable
Returns	None	
Error		

### 20-3 IF~ THEN~ ELSE

Conditional Judgment

The IF command does not work across multiple lines.

Format	IF condition is met THEN command IF condition is met THEN @label IF condition met THEN command, IF condition not met ELSE command IF condition is met THEN @label or ELSE @label	
Parameters	Conditional	Comparison between content
	If condition met	Command/Branch destination

Returns Error	If condition not met	Command/Branch destination
	None	

20-4 IF~ GOTO~ ELSE

Conditional Judgment

The IF command does not work across multiple lines.

Format	IF condition met GOTO @label	
	IF condition met GOTO @label or ELSE other command	
	IF condition met GOTO @label or ELSE @label	
Parameters	Conditional	Comparison between content
	If condition met	Command/Branch destination
	If condition not met	Command/Branch destination
Returns	None	
Error		

## 21 READ

### Read Commands

The following commands relate to reading data and related tasks:

READ, DATA, RESTORE,  
TMREAD(), DTREAD()

#### 21-1 READ

Reads DATA.

Format	READ returned variable1 [, returned variable2...] READ A,B,C READ X\$,Y\$ READ X,Y,Z,G\$	
Parameters	Obtained variable...	Variable storing information taken from DATA.(Multiple designation possible.)
Returns	None	
Error	When amount of data to be read is insufficient.	

#### 21-2 DATA

Definition of data to be read with READ command

can include a mix of alphanumeric characters.

Format	DATA number, number,... DATA "string", "string",... DATA 123,"SAMPLE",...	
Parameters	Data	Sequences of strings and numbers to be divided with ','
Returns	None	
Error		

#### 21-3 RESTORE

Changes position of DATA to be READ.

Instead of an @label, you can also use a text string variable substituted for the label name.

Format	RESTORE @label RESTORE variable \$ </color>	
Parameters	@label	Acquisition Position

Returns	None
Error	

#### 22-4 TMREAD()

Convert time string into number.

Format	TMREAD( "time string" ), HOUR, MIN, SEC	
Parameters	Time string	HH:MM:SS Format of time string
	HOUR	Variable retrieving hour
	MIN	Variable retrieving minute
	SEC	Variable retrieving second
Returns	None	
Error		

#### 22-5 DTREAD()

Convert the date string into a number.

Format	DTREAD( "date string" ), YEAR, MON, DAY	
Parameters	DATA string	Date displayed in format: YYYY/MM/DD
	YEAR	Variable retrieving Year
	MON	Variable retrieving month
	DAY	Variable retrieving day
Returns	None	
Error		

## 22 Basic Console Commands

The following commands allow you to display or adjust characters on the console screen:

CLS, COLOR, LOCATE, PRINT, CHKCHR(), ACLS, VISIBLE

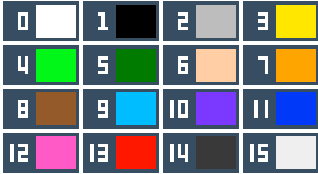
### 22-1 CLS

Erases the contents of the console screen.

Format	CLS	
Parameters	None	
Returns	None	
Error		

### 22-2 COLOR

Character color is specified on console display.

Format	TMREAD( "time string" ), HOUR, MIN, SEC	
Parameters	Character Color	0~15 (Uses the 15th color of the 16 color palette assigned to BG screens) 
	Background Color	0~15 (0=transparent)
Returns	None	
Error		

### 22-3 LOCATE

Designates the position of the character display on the console.

Format	LOCATE x coordinate, y coordinate	
Parameters	x coordinate	Horizontal coordinates (0-31) *Values outside valid range may be used.
	y coordinate	Vertical coordinates (0-23) *Values outside valid range may be used.
Returns	None	

Error	
-------	--

## 22-4 PRINT

Displays characters on the console.

Format	PRINT "string" PRINT variable PRINT variable\$ PRINT variable;variable\$;"string" PRINT "string",variable,variable \$	
Parameters	;	Used when displaying multiple elements one after the other.
	,	When displaying multiple elements one after the other, adjustments are made for TAB position
Returns	None	
Error		

## 22-5 CHKCHR()

Search for character numbers on the console.

Format	Variable = CHKCHR (x coordinate, y coordinate)	
Parameters	x coordinate	Used when displaying multiple elements one after the other.
	y coordinate	When displaying multiple elements one after the other, adjustments are made for TAB position
Returns	Number	0-255=character code (-1=outside range)
Error		

## 22-6 ACLS

Reset Graphic Display Environment

Format	ACLS
Parameters	None
Returns	None

Error	
-------	--

The program below will return settings such as console, SPRITE, BG, graphic display status and color settings to their default:

```
VISIBLE 1,1,1,1,1,1: ICONCLR
COLOR 0: CLS: GDRAWMD FALSE
FOR P=1 TO 0 STEP -1
  GPAGE P,P,P: GCOLOR 0: GCLS: GPRI0 3
  BGPAGE P: BGOFS 0,0,0: BGOFS 1,0,0
  BGCLR: BGCLIP 0,0,31,23
  SPPAGE P: SPCLR
NEXT
FOR I=0 TO 255
  COLINIT "BG", I: COLINIT "SP", I
  COLINIT "GRP",I
NEXT
```

### 22-7 VISIBLE

Control of screen display elements (using 0 will cause a particular element not to be displayed, while using 1 will display it).

Format	VISIBLE console, panel, BG0, BG1, SPRITE, graphic	
Parameters	Console	0=No display 1=Display
	Panel	0=No display 1=Display
	BG0	0=No display 1=Display
	BG1	0=No display 1=Display
	Sprite	0=No display 1=Display
	Graphic	0=No display 1=Display
Returns	None	
Error		



## 23 Console Entry Commands

The following functions and commands will return data on inputted button and strings:

INKEY\$(), INPUT, LINPUT, BUTTON(), BTRIG(), BREPEAT

### 23-1 INKEY\$()

Obtains a single character inputted on the keyboard.

This command will output the data from the TAB key converted into a space. For system reasons, Backspace will not be returned. To use these keys, utilize a keyboard system variable.

Format	Variable\$=INKEY\$0	
Parameters	None	
Returns	Character variable	A single keyboard character will be returned.(When there is nothing inputted, it will return "").)
Error		

### 23-2 INPUT

Obtain numbers or strings.

Format	INPUT ["string";] received variable INPUT ["string";] received variable\$ INPUT ["string";] received variable, received variable2\$	
Parameters	String	Explanatory text for entry
	Received variable	Text string variable or value for obtaining data entered on the keyboard. Use commas to split up commands, so you can enter multiple commands.
Returns	None	
Error		

### 23-3 LINPUT

Retrieves string, including characters like ',' which cannot be entered via INPUT.

Format	LINPUT ["string";] received variable\$	
Parameters	String	Explanatory text for entry

Returns Error	Received variable	Variable for receiving a one line string entered via keyboard.
	None	

#### 23-4 BUTTON()

This returns data from each button pressed.

When buttons are pressedd simultaneously, the data is retrieved in bit from. For instance, if up and right are pressed at the same time, the value 9 is returned. When using values beside those for buttons being pressed, use VSYNC1 to synchronize the action, completing it within 1/60<sup>th</sup> of a second during the man loop. Completing it within 1/60<sup>th</sup> of a second during the main loop.

Format	Variable=BUTTON( [type] )	
Parameters	Type (if unspecified, will be set to 0)	
	0	Pressed
	1	Instant pressed (including rapid-fire button presses)
	2	Instant pressed
	3	Instant released
Returns	Number that corresponds to button	
	1	Up on +Control Pad
	2	Down on +Control Pad
	4	Left on +Control Pad
	8	Right on +Control Pad
	16	A Button
	32	B Button
	64	X Button
	128	Y Button
	256	L Button
	512	R Button
	1024	START
Error		

#### 23-5 BTRIG()

This returns data from the instant a button is pressed.

To use a precise value, use VSYNC1 to synchronize the a action, completing it within 1/60<sup>th</sup> of a second during the main loop.

Format	Variable=BTRIG()	
Parameters	None	
Returns	Variable	Variable corresponding to button * <a href="#">BUTTON()</a> <a href="#">Reference</a>
Error		

## 23-6 BREPEAT

### Data Settings for Repeated Button Presses

The multiple button press function is usually set to OFF. Use this command to activate it. When you designate a button using the Button ID, that button's repeated press function will be switched OFF as standard. The unit of time 1 corresponds to 1/60th of a second.

Format	BREPEAT Button ID [ , start time, interval ]	
Parameters	Button ID (control number)	
	0	Up on +Control Pad
	1	Down on +Control Pad
	2	Left on +Control Pad
	3	Right on +Control Pad
	4	A Button
	5	B Button
	6	X Button
	7	Y Button
	8	L Button
	9	R Button
	10	START
	Start Time	0 -
	Interval	1 - (0=Pause)
Returns	None	
Error		

## 24 Panel & Icon Commands

The following commands let you perform tasks like changing panel type, and checking user system icon status or adjusting their settings:

**PNLTYPE, PNLSTR, ICONSET, ICONCLR, ICONCHK()**

### 24-1 PNLTYPE

Changes the panel type.

Format	PNLTYPE "panel name"	
Parameters	Panel name	
	Strings that select type to be displayed on Lower Screen:	
	OFF	No panel is displayed
	PNL	When there is no keyboard
	KYA	English keyboard
	KYM	Symbol keyboard
	KYK	Kana keyboard
Returns	None	
Error		

### 24-2 PNLSTR

String Display on Lower Screen.

Unlike on the console on the upper screen, line breaks will not be added automatically, even when displaying the last line.

Format	PNLSTR x coordinate, y coordinate, "string", palette number	
Parameters	x coordinate	Horizontal coordinates (0-31) *Values outside valid range may be used.
	y coordinate	Vertical coordinates (0-23) *Values outside valid range may be used.
	Palette number	0 – 15
Returns	None	
Error		

### 24-3 ICONSET

Settings for user system icon characters (or to initiate display).

Format	ICONSET icon position, icon number	
Parameters	Icon position	User system icon number (0 – 3)
	Icon number	Character control numbers for icons )0- 63)
Returns	None	
Error		

#### 24-4 ICONCLR

Cancels display of user system icons.

Format	ICONCLR icon position	
Parameters	Icon position	0 – 3 (if unspecified, this will apply to all icons)
Returns	None	
Error		

#### 24-5 ICONCHK()

Check current status of user system icons.

Format	Number = ICONCHK()	
Parameters	None	
Returns	Number	0 – 3 (icon location), -1 = not pressed
Error		

## 25 File & Communication Commands

The following commands are used for operations relating to files, such as loading, saving and deleting. Dialog boxes will appear after using these commands, allowing you to confirm your decision:

**LOAD, SAVE, DELETE, RENAME, RECVFILE, SENDFILE**

### 25-1 LOAD

Loads file.

Format	LOAD "resource name:file name" [ , display control ]	
Parameters	Resource Name	
	Strings assigned to the resources to be read.	
	PRG	Program (Can be omitted)
	MEM	Memory
	COL0	BG Colors
	COL1	SPRITE Collors
	COL2	Graphic Colors
	GRP0 : GRP3	Graphics (For 4 Pages)
	SCU0 SCU1	Foreground layer Background layer
	BGU0 : BGU3	User's GB Characters(Bank x 4)
	SPU0 : SPU3	User Sprite Characters (Bank x 8)
Returns	Display control	Enter FALSE and the dialog box will not be displayed during loading.
	None	
Error	Result	FALES TRUE CANCEL

### 25-2 SAVE

This saves a file. (A dialog box confirming the decision will appear.)

You cannot use filenames that match the names of files included as samples with this software.

Format	SAVE "Resource name:file name"	
Parameters	Resource name	<a href="#">*Refer to LOAD</a>
Returns	None	
Error	RESULT	FALES TRUE CANCEL

### 25-3 DELETE

Erases file.

You cannot delete files included as samples with this software.

Format	DELETE "Resource name:file name"	
Parameters	Name of File Type	<a href="#">*Refer to FILES</a>
Returns	None	
Error	RESULT	FALES TRUE CANCEL

### 25-4 RENAME

Changes file names.

Format	RENAME "resource name: file name", "new name"	
Parameters	Name of File Type	<a href="#">*Refer to FILES</a>
Returns	None	
Error	RESULT	FALES TRUE CANCEL

### 25-5 RECVFILE

Receive a Petit Computer file another user has saved on their DSi system (displays confirmation message).

Format	RCVFILE "resource name: file name"	
Parameters	Name of File Type	<a href="#">*Refer to FILES</a>
Returns	None	

Error	RESULT	FALES TRUE CANCEL
-------	--------	-------------------------

## 25-6 SENDFILE

Send files to another user who has a DSi with Petit Computer saved on it.

Format	SENDFILE "resource file: file name"	
Parameters	File type name	<a href="#">*Refer to FILES</a>
Returns	None	
Error	RESULT	FALES TRUE CANCEL



## 26 File Commands (Expert)

These commands are geared towards experts who have a high level of understanding of files and resources. If you do not understand these commands after reading the explanation, please avoid using them.

**APPEND, EXEC, SAVE (as package)**

### 26-1 APPEND

Combining Programs (In Run Mode)

You can add another program to the end of a program you are editing. If you use the APPEND command without having fully understood how the command functions, you may append a program to the one you are editing and prevent your current program from working properly. Please ensure that you have fully understood the command before making use of it.

Format	APPEND "File Name"	
Parameters	File name	File names of programs you wish to combine.
Returns	None	
Error	RESULT	FALES=Failure

### 26-2 EXEC

Loads and runs other programs from within the current program.

Format	EXEC "file name"	
Parameters	File name	Name of program file to run
Returns	None	
Error	RESULT	FALES=Failure

### 26-3 Save (As Package)

This allows you to save programs and resources together.

Files outputted in package form can be extremely large. Ensure that you have enough free space available when saving programs as package-type files. Please note that when programs saved as package-type files are opened with a LOAD command, all resources included in the package will also be loaded.

Format	SAVE "Resource Name:File Name", "Package Parameter String"	
Parameters	Package Parameter String	Data for Designating Resources to be Saved at Same Time
Returns	None	

Error	RESULT	FALES TRUE CANCEL
-------	--------	-------------------------

### 26-4 Package Parameter String

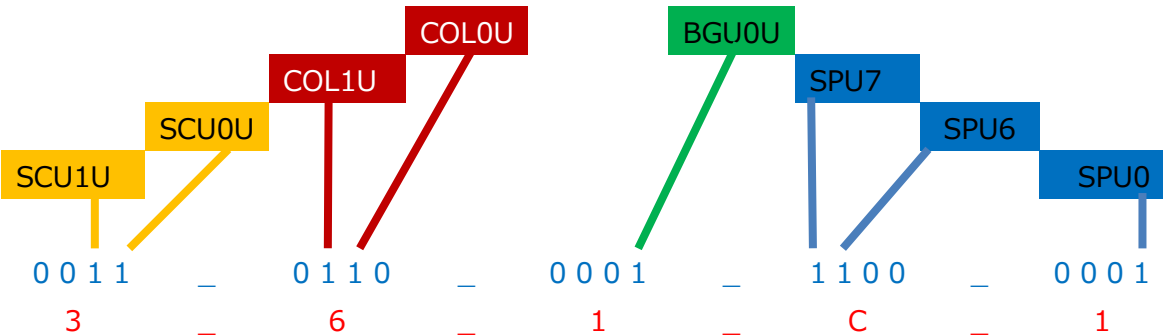
Bitwise data allowing you to select resource types will be displayed in hexadecimal string form.

Change the bitwise data corresponding to resources saved at the same time to 1 and you can save data including all resources in a single file.

(e.g.) You can save resources such as these:

- Upper screen user's sprite 0, 6, 7 and color
- Both upper screen user's BG0 and BG Screen and color

The resources structures above can be expressed in binary or hexadecimal form.



Save package parameters as a hexadecimal string.

SAVE "TEST", "361C1"

Please note that when using HEX\$, a maximum value of 20 bits can be used.

b00	SPU0 Upper Screen User Sprite Character 0
b01	SPU1 Upper Screen User Sprite Character 1
b02	SPU2 Upper Screen User Sprite Character 2
b03	SPU3 Upper Screen User Sprite Character 3
b04	SPU4 Upper Screen User Sprite Character 4
b05	SPU5 Upper Screen User Sprite Character 5
b06	SPU6 Upper Screen User Sprite Character 6
b07	SPU7 Upper Screen User Sprite Character 7
b08	BGU0U Upper Screen User BG Character 0
b09	BGU1U Upper Screen User BG Character 1
b10	BGU2U Upper Screen User BG Character 2

b11	BGU3U Upper Screen User BG Character 3
b12	BGUFU Upper Screen Font
b13	COL0U Upper Screen BG Color
b14	COL1U Upper Screen SPRITE Color
b15	COL2U Upper Screen Graphic Color
b16	SCU0U Upper Screen BG Foreground Layer
b17	SCU1U Upper Screen BG Background Layer
b18	GRP0 Graphic Page 0
b19	GRP0 Graphic Page 1
b20	GRP0 Graphic Page 2
b21	GRP0 Graphic Page 3
b22	MEM Memory String
b23	System Reservation
b24	System Reservation
b25	System Reservation
b26	System Reservation
b27	BGU0L Lower Screen User BG Character 0
b28	BGU1L Lower Screen User BG Character 1
b29	BGU1L Lower Screen User BG Character 2
b30	BGU1L Lower Screen User BG Character 3
b31	BGUFL Lower Screen Font
b32	COL0L Lower Screen BG Color
b33	COL1L Lower Screen SPRITE Color
b34	COL2L Lower Screen Graphic Color
b35	SCU0L Lower Screen BG Foreground Layer
b36	SCU1L Lower Screen BG Background Layer
b37	System Reservation
b38	System Reservation
b39	System Reservation
b40	System Reservation
b41	System Reservation

b42	System Reservation
b43	System Reservation
b44	System Reservation
b45	Unused
b46	Unused
b47	Unused

## 27 Basic Mathematical Functions

The following mathematical functions allow you to perform tasks including obtaining integers, absolute values and codes and generating random numbers:

**FLOOR(), RND(), ABS(), SGN()**

### 27-1 FLOOR()

Obtain the integer or whole number.

You can also use the AND command to obtain an integer in a 1 byte range (e.g.) A=A AND &HFF

Format	Variable = FLOOR( number )	
Parameters	Number	Number
Returns	Number	Requested result
Error		

### 27-2 RND()

Gives a random number up to the designated value.

Format	Variable = RND(maximum number )	
Parameters	Maximum Number	Maximum number generated
Returns	Number	Random number from 0 – maximum number (not including the maximum value)
Error		

### 27-3 ABS()

Obtains an absolute value.

Format	Variable = ABS( number )	
Parameters	Number	Number from which you want to obtain an absolute value
Returns	Number	Absolute value
Error		

### 27-4 SGN()

Obtains a code.

Format	Variable = SGN( variable )	
Parameters	Number	Number to check code
Returns	Number	0 or +1,-1

Error	
-------	--

## 28 Exponential Figures & Logarithms

"The following commands allow you to perform logarithms and generate exponential figures (numbers to the power of other numbers):

**SQR()**, **EXP()**, **LOG()**, **POW()**

### 28-1 SQR()

Obtains the square root of a number.

Format	Variable = SQR( number )	
Parameters	Number	Original numerical value
Returns	Number	Requested result
Error	When value is negative number	

### 28-2 EXP()

Looks for the exponent value.

Format	Variable = EXP( number )	
Parameters	Number	Original numerical value
Returns	Number	Requested result
Error		

### 28-3 LOG()

Calculated natural logarithm.

Format	Variable = LOG( variable )	
Parameters	Number	Original numerical value
Returns	Number	Calculated result
Error		

### 28-4 POW()

Use this command to return an exponential value - multiplying a number by the power of another number.

Format	Variable=POW(numerical value, exponential value)	
Parameters	Number	Original numerical value
Returns	Exponential value	Value when multiplied to power of exponential value
Error	When the value is a negative number and the exponential value is not a whole number (integer).	





## 29 Trigonometric Functions

The following mathematical functions allows you to perform calculations including the value of sine and cosine:

PI(), RAD(), DEG(), SIN(), COS(), TAN(), ATAN()

### 29-1 PI()

Obtains value of PI.

Format	Variable=PI()	
Parameters	None	
Returns	Number	Value of PI (circumference ratio)
Error		

### 29-2 RAD()

Obtain a radian figure from angle data.

Format	Variable=RAD(angle)	
Parameters	Angle	0 – 360
Returns	Number	Radian figure from angle
Error		

### 29-3 DEG()

Obtains angle data from radian value.

Format	Variable=DEG(radian)	
Parameters	Radian	0 – 2π
Returns	Number	Angle from radian value
Error		

### 29-4 SIN()

Returns sine value.

Format	Variable=SIN(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result
Error		

### 29-5 COS()

Returns cosine value.

Format	Variable=COS(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result
Error		

#### 29-6 TAN()

Returns tangent value.

Format	Variable=TAN(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result
Error		

#### 29-7 ATAN()

Obtains the arc tangent value.

Can also be used as a function for determining direction from 2 parameters ( Y, X ) and displacement. Desired direction=ATAN( destination y-y, destination x-x )

Format	Variable=ATAN(radian)	
Parameters	Radian	Radian value of angle
Returns	Number	Requested result
Error		

### 31 Basic Character Functions

The following character-related functions allow you to obtain data from strings and find the codes for designated characters:

ASC(), CHR\$(), VAL(), STR\$(), HEX\$()

#### 30-1 ASC()

Character code of designated character.

Format	Variable=ASC(character)	
Parameters	Character	Single Character
Returns	Number	Character code of designated character
Error		

#### 30-3 VAL()

Obtains a number from a string.

Format	Variable=VAL(string)	
Parameters	String	String with number
Returns	Number	Number extracted from string
Error		

#### 30-4 STR\$()

Obtain a string from a number.

Format	Variable=STR\$(number)	
Parameters	Number	Number you want to convert to string
Returns	Character	String generated from number
Error		

#### 30-5 HEX\$()

Gives a hexadecimal string from a number.

Format	Variable\$ = HEX\$ (numerical value [, decimal places])	
	Number	Number you want to convert to hexadecimal string
	Decimal Places	1~5 (When the value does not correspond to the given number of decimal places, the remaining

Returns		figures will be replaced by 0.)
	Character	Hexadecimal string generated from number.
Error		

### 31 Search & Replace Character Functions

The following functions and commands will return data on character numbers, and obtain sections of strings:

LEN, MID\$, RIGHT\$, LEFT\$, INSTR(), SUBST\$

#### 31-1 LEN()

Obtains the number of characters within a string.

Format	Variable=LEN(string)	
Parameters	String	String you want to determine the length of
Returns	Number	Number of characters (every character is counted as 1)
Error		

#### 31-2 MID\$()

Extracts a string of a designated length starting from the initial position within the target string.

Format	Variable\$ = MID\$( string, initial position, number of characters )	
Parameters	String	Original string
	Initial position	Initial position of characters
	Number of characters	Number of characters to be retrieved
Returns	Character	Extracted string
Error		

#### 31-3 RIGHT\$()

The designated number of characters will be obtained, counting from the right side of the character string.

Format	Variable\$ = RIGHT\$ ( character string, number of characters )	
Parameters	Number of characters	Original string
	Number of characters	Number of characters to be retrieved
Returns	Character	Extracted string
Error		

#### 31-4 LEFT\$()

The designated number of characters will be obtained, counting from the left side of the character string.

Format	Variable\$ = LEFT\$ ( character string, number of characters )	
Parameters	String	Original string
	Number of characters	Number of characters to be retrieved
Returns	Character	Extracted string
Error		

### 31-5 INSTR

This will search for a particular character string within the designated search range.

Format	Variable = INSTR ( character string, character strings to search )	
Parameters	Number of characters	Original string
	Character strings to search within	Character string to search for
Returns	Number	Position within character string (0~, -1=None)
Error		

### 31-6 SUBST\$

Use this command to replace a character string.

Format	Variable \$ = SUBST\$ ( character string, start position, number of characters, character string to substitute with )	
Parameters	String	Original string
	Initial position	Original position of the string you wish to replace (0~number of characters -1)
	Number of characters	Number of characters to replace
	Modified character string	Modified character string
Returns	Character	Character string after replacement.
Error		

## 32 Basic Graphic Commands

The following commands perform tasks such as designating the graphic page to be used and erasing graphics:

**GPAGE, GCOLOR, GCLS, GSPOIT()**

### 32-1 GPAGE

Designates the graphic screen to be used.

Format	GPAGE screen	
Parameters	Screen	0=Upper Screen 1=Lower Screen
Returns	None	
Error		

### 32-2 GCOLOR

Assigns graphic color on graphic screen.

Format	GCOLOR color number	
Parameters	Color number	0 - 255
Returns	None	
Error		

### 32-3 GCLS

Erases images on designated graphic screen.

Format	GCLS [ color ]	
Parameters	Color	0 - 255
Returns	None	
Error		

### 32-4 GSPOIT()

Checks color of designated location.

Format	Variable=GSPOIT (x coordinate, y coordinate)	
Parameters	x coordinate	0 - 255 (Values outside valid range may be used)
	y coordinate	0 - 191 (Values outside valid range may be used)
	Color	0 - 255 (-1 if outside range)

Returns	None
Error	



### 33 Graphic Screen Commands

The following commands allow you to draw lines and circles and color sections of the screen:

GPSET, GPAINT, GLINE, GBOX, GFILL, GCIRCLE, GPUTCHR

#### 33-1 GPSET

Adds a dot.

Format	GPSET x coordinate, y coordinate [ ,color ]	
Parameters	x coordinate	0 – 255 (Values outside valid range may be used)
	y coordinate	0 – 191 (Values outside valid range may be used)
	Color	0 – 255 (-1 if outside range)
Returns	None	
Error		

#### 33-2 GPAINT

Fills in color from designated point.

Where the border color has been designated, the area this border surrounds will be filled with a single color. To save time, any color adjacent to the designated location which has the same color will be filled in. This does not work in XOR display mode.

Format	GPAINT x coordinate, y coordinate [ ,color ]	
Parameters	x coordinate	0 – 255 (Values outside valid range may be used)
	y coordinate	0 – 191 (Values outside valid range may be used)
	Color	0 – 255 (-1 if outside range)
	Border Color	0 – 255
Returns	None	
Error		

#### 33-3 GLINE

Draws a line.

Format	GLINE x start point, y start point, x end point, y end point [ ,color ]	
Parameters	x start point	0 – 255 (Values outside valid range may be used)

	y start point	0 – 191 (Values outside valid range may be used)
	x end point	0 – 255 (Values outside valid range may be used)
	y end point	0 – 191 (Values outside valid range may be used)
	Color	0 – 255 (if omitted, color will be GCOLOR)
	Returns	None
Error		

### 33-4 GBOX

Draws a box.

Format	GBOX x start point, y start point, x end point, y end point [ ,color ]	
Parameters	x start point	0 – 255 (Values outside valid range may be used)
	y start point	0 – 191 (Values outside valid range may be used)
	x end point	0 – 255 (Values outside valid range may be used)
	y end point	0 – 191 (Values outside valid range may be used)
	Color	0 – 255 (if omitted, color will be GCOLOR)
Returns	None	
Error		

### 33-5 GFILE

Fills in color of a rectangle.

Format	GFILE x start point, y start point, x end point, y end point [ ,color ]	
Parameters	x start point	0 – 255 (Values outside valid range may be used)
	y start point	0 – 191 (Values outside valid range may be used)
	x end point	0 – 255 (Values outside valid

		range may be used)
	y end point	0 – 191 (Values outside valid range may be used)
	Color	0 – 255 (if omitted, color will be GCOLOR)
	Returns	None
Error		

### 33-6 GCIRCLE

Draws a circle.

Format	GCIRCLE x coordinate, y coordinate, radius [ ,color ] [, initial angle, final angle ]	
Parameters	x coordinate	0 – 255 (Values outside valid range may be used)
	y coordinate	0 – 191 (Values outside valid range may be used)
	Radius	0 – 255 (Values outside valid range may be used)
	Color	0 – 255 (if omitted, color will be GCOLOR)
	Initial angle	0 – 360 (Values outside valid range may be used)
	Final angle	0 – 360 (Values outside valid range may be used)
Returns	None	
Error		

34 Graphic Commands (Expert)

The following commands allow you to perform tasks such as designating special pages, copying screens, and displaying characters on the graphic screen:

GPAGE, GDRAWMD, GPRIO, GCOPY, GPUTCHR

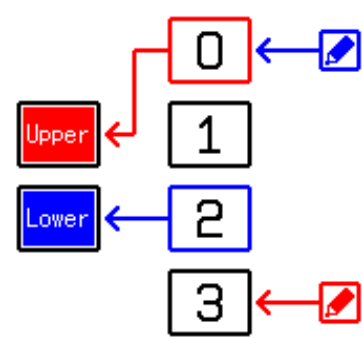
34-1 GPAGE(Expert)

This allows you to select a graphic screen to control, and then demarcate internal graphic areas within it, allocating specific areas for drawing and display.

Initialized situation and Run ACLS Command function, Assignment GPAGE 0,0,0 and GPAGE 1,1,1.

Format	GPAGE screen [,drawing page , display page]	
Parameters	Screen	0=Upper Screen 1=Lower Screen
	Drawing Page	0 – 3
	Display Page	0 – 3
Returns	None	
Error		

On the upper and lower screen, you can allocate the pages to be displayed from a total of 4. First, select either the upper or lower screen and then allocate the pages you wish to have displayed on it. You can use drawing commands such as GPAINT to draw on a page that has not been selected for on-screen display. When you have finished the graphics, it is possible to then switch to this screen when you wish to have it displayed.



34-2 GPRIO

Modifying Display Order Priority on the Graphic Screen

As standard, the display always appears behind the sprites on screen. The values for the display order priority are designed for sprite display.

Format	GPRIO Number	
Parameters	Number	0 – 3
Returns	None	
Error		

### 34-3 GDRAWMD

This is used to designate drawing color in XOR Display Mode.

When using the XOR display mode, you can erase the contents of an image by drawing an image twice in the same place.

Format	GDRAWMD Status	
Parameters	Status	FALSE=Normal color TRUE=XOR
Returns	None	
Error		

### 34-4 GCOPY

This command copies the graphic screen.

Format	GDRAWMD StatusGCOPY [ transfer source page, ] start x, start y, end x, end y, destination x, destination y, Copy Mode	
Parameters	x coordinate	0 – 255 (Values outside valid range may be used)
	y coordinate	0 – 191 (Values outside valid range may be used)
	Character name	*Refer to CHRINIT
	Character number	Character color (1 – 15)
	Scale	Magnification (1,2,4,8)
Returns	None	
Error		

### 34-5 GPUTCHR

This displays the assigned character graphic data on the graphic screen.

This command will copy the data for the designated palette number to the graphic palette. The assigned location will be colored with the 16th shade of the 16th color from the designated palette number.

Format	GPUTCHR x coordinate, y coordinate, "character name", number,
--------	---

Parameters	palette number, scale	
	Transfer Source Page	0 – 3
	Start point x	0 – 255 (Values outside valid range may be used)
	Start point y	0 – 191 (Values outside valid range may be used)
	End point x	0 – 255 (Values outside valid range may be used)
	END point y	0 – 191 (Values outside valid range may be used)
	Transfer destination x	0 – 255
	Transfer destination y	0 – 191
	Copy mode	Copy of Color 0 FALES = Do not perform True = Perform
Returns	None	
Error		

### 35 Color & Character Commands

The following commands allow you to control the elements displayed on the screen, designate colors or retrieve data, define characters or reset the screen:

COLINIT, COLSET, COLREAD(), CHRINIT, CHRSET, CHRREAD()

#### 35-1 COLINT

Restores the initial color.

Format	COLINIT "color bank name", color number	
Parameters	Color bank name	
	Strings designating target:	
	BG	BG screens
	SP	Sprites
	GRP	Graphics
	Color number	0 – 255
Returns	None	
Error		

#### 35-2 COLSET

Assign new color

BG number 0 is the background color

Format	COLSET "color bank name", color number, "color data string"	
Parameters	Color bank name	*Refer to COLINIT
	Color number	*Refer to COLINIT
	Color data string	In hexadecimal (base 16) notation(the order is RRGGBB) Each element will be 00~FF (e.g.)"FF00AA"
Returns	None	
Error		

#### 35-3 COLREAD()

Retrieves designated color data. (Each element 0~255)

Format	COLREAD( "color bank name", color number ), R, G, B	
Parameters	Color bank name	*Refer to COLINIT
	Color number	*Refer to COLINIT

Returns Error	R	Variable for red
	G	Variable for green
	B	Variable for blue
	None	

#### 35-4 CHRINIT

Resets designated character to initial state.

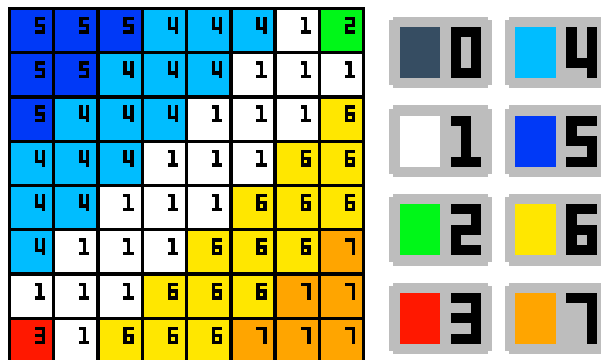
Format	CHRINIT "character name"	
Parameters	Character name	
	Strings designating character:	
	BGU0 : BGU3 SPU0 : SPU7	User BG character
		User sprite character
Returns	None	
Error		

#### 35-5 CHRSET

Define a single character (8x8 pixel units)

Format	CHRSET "character name", character number, "graphic string"	
Parameters	Character name	*Refer to <a href="#">CHRINIT</a>
	Character Number	0 – 255
	Graphic string	16 color 8x8 pixel character data is expressed in hexadecimal (base 16) notation
Returns	None	
Error		





(e.g.) A hexadecimal string has been generated from the character shown above:  
 "5554441255444111544411164441116644111666411166671116667731666777"  
 (each character represents 1 pixel)

### 35-6 CHRREAD()

Retrieves data for the designated character.

Format	CHRREAD( "character name", character name ), C\$	
Parameters	Character name	<a href="#">*Refer to CHRINIT</a>
	Character Number	<a href="#">*Refer to CHRINIT</a>
	C\$	Variable for graphic string <a href="#">*Refer to CHRSET</a>
Returns	None	
Error		

## 36 Basic Sprite Commands

The following commands allow you to perform actions such as starting and pausing sprite movement:

SPPAGE, SPSET, SPCLR, SPHOME

### 36-1 SPPAGE

Designates the screen to be used for sprites.



Although the lower screen can be selected, it is generally used for the keyboard. User characters cannot be displayed on this screen, and only the simple graphics already pre-loaded can be used.

Format	SPPAGE screen	
Parameters	Screen	0 = Upper Screen 1 = Lower Screen
Returns	None	
Error		

### 36-2 SPSET

Sprite definition (activation).

Activates a sprite designated by a control number. The coordinates are reset to 0,0. Once a sprite has been activated using SPSET and you wish only to change the character number, use the SPRCHR command.

Format	SPSET control number, character number, palette number, horizontal rotation, vertical rotation, order of precedence		
Parameters	Control number	0 – 99	
	Sprite character number	0~511 (for display on lower screen 0~117)	
	Palette number	0~15	
	horizontal rotation	0=None 1=Horizontal rotation 	
	Vertical rotation	0=None 1=Vertical rotation 	
	Order of Priority	0	In front of console

		<table><tr><td>1</td><td>In front of BG (front layer)</td></tr><tr><td>2</td><td>Between 2 BG layers</td></tr><tr><td>3</td><td>Behind rear BG layer</td></tr></table>	1	In front of BG (front layer)	2	Between 2 BG layers	3	Behind rear BG layer
	1	In front of BG (front layer)						
	2	Between 2 BG layers						
	3	Behind rear BG layer						
		The order of priority for sprite display is determined by the control number: the sprite with the lower number will be displayed further forward.						
Width	8, 16, 32, 64 (Unless specified, this value will be 16)							
height	8, 16, 32, 64 (Unless specified, this value will be 16)							
Returns	None							
Error								

The following combinations of height and width values cannot be assigned: 8x64, 16x64, 64x8, 64x16

### 36-3 Rules for Storing Sprites

When the SPSET command has been used to assign sprite size, attention should be paid to the size and the character number of the displayed sprite. A character defined in the normal way will be stored as a unit of 16x16 pixels. If a character of a different size is used, its positioning will have to be adjusted as required.

0	0 1	0 1 2 3	64x8(Not Permitted)
8x8	16x8	32x8	
0	0 1	0 1 2 3	64x16(Not Permitted)
1	2 3	4 5 6 7	
8x16	16x16	32x16	
0	0 1	0 1 2 3	0 1 2 3 4 5 6 7
1	2 3	4 5 6 7	8 9 10 11 12 13 14 15
2	4 5	8 9 10 11	16 17 18 19 20 21 22 23
3	6 7	12 13 14 15	24 25 26 27 28 29 30 31
8x32	16x32	32x32	64x32



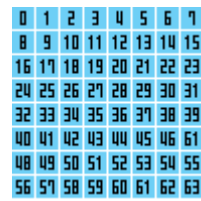
8x64 (Not Permitted)



16x64 (Not Permitted)



32x64



64x64

### 36-4 SPCLR

Erase sprite (Prevent sprites being activated)

Format	SPCLR control number	
Parameters	Control number	0~99 (if omitted, all sprites will be erased)
Returns	None	
Error		

### 36-5 SPHOME

Designating Start Point for Sprite Display

When this command is omitted, the start point for the display will be assigned as top left (0,0).

Format	SPHOME control number, x, y	
Parameters	Control number	0~99
	x	0~63
	y	0~63
Returns	None	
Error		

### 37 Sprite Control Commands

The following commands allow you to change sprite coordinates or animate sprites:

SPOFS, SPCHR, SPANIM, SPANGLE, SPSCALE

#### 37-1 SPOFS

Changes sprite coordinates.

Format	SPOFS control number, x coordinate, y coordinate [,interpolation time]	
Parameters	x coordinate	±1024 (Values outside valid range may be used)
	y coordinate	±1024 (Values outside valid range may be used)
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60 <sup>th</sup> sec)
Returns	None	
Error		

#### 37-2 SPCHR

Changes the sprite character number.

Format	SPCHR control number, character number [, palette number, horizontal rotation, vertical rotation, order of precedence ]	
Parameters	Control number	0 – 99
	Sprite character number	0 – 511 (for display on lower screen 0 – 117
	Palette number	0 – 15
	Horizontal rotation	0=none, 1=rotate
	Vertical rotation	0=none, 1=rotate
	Order of precedence	0 – 3
Returns	None	
Error		

#### 37-3 SPANIM

Displays sprite animation.

Starting with the current designated character number, using this command will make the character number change at defined intervals, It will change within the range of the designated number of frames.

Format	SPANIM control number, number of frames, time [, loop]	
Parameters	Control number	0 – 99
	Number of frames	1~
	Time	Time to display 1 frame (1=1/60th sec)
	Loop	0=Endless loop, 1~ (Loop number)
Returns	None	
Error		

#### 37-4 SPANGLE

This is used to modify the angle of sprites.

From the initial position, the start point for rotation will be at the top left of the sprite.

To modify the point around which the sprite rotates, use the SPHOME command.

Format	SPANGLE control number, angle [, interpolation time, change direction ]	
Parameters	Control number	0 – 31
	Angle	0 – 360 (Values outside valid range may be used)
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60th sec)
	Change direction	1=Clockwise -1=Anticlockwise (if omitted, will be clockwise)
Returns	None	
Error		

#### 37-5 SPSCALE

Changes the scaling of sprites.

When using SPANGLE to rotate a sprite with a scale of 200%, any section of the sprite that protrudes beyond the rectangular area corresponding to this 200% will not be displayed.

Format	SPSCALE control number, scale [, interpolation time ]	
Parameters	Control number	0 – 31
	Scale	0 – 200 (proportion in percent)
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60th sec)
Returns	None	
Error		

### 38 Commands for Obtaining Sprite Data

The following commands allow a whole range of sprite data to be obtained:

**SPCHK(), SPREAD(), SPSETV, SPGETV()**

#### 38-1 SPCHEK()

Automatically retrieves interpolation data.

Format	Variable = SPCHK( control number)		
Parameters	Control number	0 – 99	
Returns	Number	Interpolation has ended when value reaches 0.	
		b00	Coordinate
		b01	Angle
		b02	Scale
		b03	Play Animation
Error			

#### 38-2 SPREAD()

This command is used to read sprite data.

**This allows you to obtain data relating to angles and new coordinates when interpolation is used.**

Format	SPREAD (control number), X, Y [, A ], [ S ], [ C ]	
Parameters	Control number	0 – 99
	Scale	0 – 200 (proportion in percent)
	X	Variable for obtaining X coordinate
	Y	Variable for obtaining Y coordinate
	A	Variable for obtaining angle
	S	Variable for obtaining scale
	C	Variable for obtaining character data
Returns	None	
Error		

#### 38-3 SPSETV



### Pre-set Variable Assigned to Each Sprite

Users are free to use these variables as they like. For instance, each sprite could retain data on health, attack and defense power without needed to program an array. Use these variables by substituting them for the initial values.

Format	SPSETV control number, variable number, value	
Parameters	Control number	0 – 99
	Variable number	0 – 7
	Value	Number
Returns	None	
Error		

### 38-4 SPGETV()

This is used to read the pre-set variables assigned to each sprite.

Format	Variable = SPGETV( control number, variable number )	
Parameters	Control number	0 – 99
	Variable number	0 – 7
Returns	Number	Variable control
Error		

## 39 Sprite Collision Detection Commands

The following commands relate to collision detection for sprites:

**SPCOL, SPCOLVEC, SPHIT(), SPHITSP(), SPSPHITRC()**

### 39-1 SPCOL

This defines blocks for sprite collision detection.

If this command is not utilized, the offset will be 0,0 and the size will be that defined by the SPSET command.

Format	SPCOL control number, x, y width, height, scale adjustment [, group ]	
Parameters	Control number	0 – 99
	x	±64
	y	±64
	Width	1 – 64
	Height	1 – 64
	Scale Adjustment	FALSE=Ignore TRUE=Synchronize

Returns Error	Group	0~255 (Group settings in bit units)
	None	

You can allocate bits as you like within the group. For instance, the 4 lower bits could be player data, while the 4 upper bits could be enemy data. For collision detection between players and enemies, insert an &HF0 into the group.

b00	Player
b01	Player's Shots
b02	Effect
b03	Item
b04	Enemy
b05	Enemy's Shots
b06	Boss
b07	Background Obstacles

### 39-2 SPCOLVEC

This defines movement speed for use in collision detection.

This is used when the amount of movement (displacement) needs to be defined. When this is not required, the displacement can be omitted. When this command is not run, it will be automatically calculated using the time assigned with SPOFS and the value obtained from the destination. When movement time is not assigned with SPOFS, the displacement will be given as 0.

Format	SPCOLVEC control number, displacement x displacement y	
Parameters	Control number	0 – 99
	Displacement x	±16.0
	Displacement y	±16.0
Returns	None	
Error		

### 39-3 SPHIT()

This detects sprite collision.

This command is used to detect collision between sprites from the same group. When sprites collide, data from the other sprite will be preserved as system variables.

Format	SPHIT( control number, [ initial control number ] )	
Parameters	Control number	0 – 99
	Initial control number	0~99 (when omitted, it will be all numbers excluding your own)
Returns	Results	TRUE = Collision
Error		

#### 39-4 SPHITSP()

This detects collision between sprites.

This command is used to detect collision between the user's sprites, and those of the designated opponent's sprites. Following collision, the opponent's sprite data will be retained as system variables.

Format	SPHITSP ( control number, opponent control number )	
Parameters	Control number	0 – 99
	Opponent control number	0 – 99
Returns	Results	TRUE = Collision
Error		

#### 39-5 SPHITRC()

This detects collision between sprites and blocks.

This command detects collision between sprites and blocks of the size designated by you. Following collision, the sprite data of the sprite defined as the opponent will be retained as system variables.

Format	SPHITRC ( control number, start x, start y, width, height [ , displacement x, displacement y ] )	
Parameters	Control number	0 – 99
	x start point	±1024
	y start point	±1024
	Width	1 –
	Height	1 –
	Displacement x	±16.0
	Displacement y	±16.0
Returns	Results	TRUE = Collision
Error		



## 40 BG Basic Commands

The following commands allow you to perform tasks such as designating the BG screen to be controlled, altering display offsets and writing onto assigned BG screen positions:

**BGPAGE, BGCLIP, BGOFS, BGPUR, BGRAD()**

### 40-1 BGPAGE

Designates the BG screen to be controlled.

Format	BGPAGE screen	
Parameters	Screen	0=Upper Screen 1=Lower Screen
Returns	None	
Error		

### 40-2 BGCLR

Clears the BG screen (filling it with character 0)

Format	BGCLR [ layer ]	
Parameters	Layer	0=Foreground,1=Background (Applies to both if not specified)
Returns	None	
Error		

### 40-3 BGCLIP

Assigns display parameters (for all layers).

Format	BGCLIP x start point, y start point, x end point, y end point	
Parameters	x start point	0~31
	y start point	0~23
	x end point	0~31
	y end point	0~23
Returns	None	
Error		

### 40-4 BGOFS

Alters offset of BG screen display.

Format	BGOFS layer, x coordinate, y coordinate [ , interpolation time ]
--------	--

Parameters	Layer	0=Foreground 1=Rear
	x coordinate	Horizontal coordinate (0 – 511) (Values outside valid range may be used)
	y coordinate	Vertical coordinate (0 – 511) (Values outside valid range may be used)
	Interpolation time	Time taken to automatically add interpolation between current state and new value. (1=1/60th sec)
Returns	None	
Error		

#### 40-5 BGPUP

Writes onto designated location on BG screens.

Format	BGPUP layer, x coordinate, y coordinate, character number, palette number, horizontal rotation, vertical rotation	
Parameters	Layer	0=Foreground 1=Rear
	x coordinate	Horizontal coordinate (0 – 63) (Values outside valid range may be used)
	y coordinate	Vertical coordinate (0 – 63) (Values outside valid range may be used)
	Character number	0 – 1023
	Palette number	0 – 15
	Horizontal rotation	0=None, 1=rotation
	Vertical rotation	0=None, 1=rotation
Returns	None	
Error		

#### 40-6 BGFILE

This command fills the BG screen within assigned block limitations.

Format	BGFILL layer, start x, start y, end x, end y, character number, palette number, horizontal rotation, vertical rotation	
Parameters	Layer	0=Foreground 1=Rear
	x start point	0 – 63
	y start point	0 – 63
	x end point	0 – 63
	y end point	0 – 63
	Character number	0 – 1023
	Palette number	0 – 15
	Horizontal rotation	0=None, 1=rotation
	Vertical rotation	0=None, 1=rotation
Returns	None	
Error		

#### 40-7 BGREAD()

Obtains data from designated location on BG screens.

Format	BGREAD( layer, x coordinate, y coordinate ), CHR, PAL, H, V	
Parameters	Layer	0=Foreground 1=Rear
	x coordinate	Horizontal coordinate (0 – 63) (Values outside valid range may be used)
	y coordinate	Vertical coordinate (0 – 63) (Values outside valid range may be used)
	CHR	Variable for character number
	PAL	Variable for palette number
	H	Variable for horizontal rotation
	V	Variable for vertical rotation

Returns	None
Error	



## 41 BG Commands (Expert)

The following commands can be used to obtain data on BG status, draw special shapes, or fill the screen with color:

**BGCHK(), BGCOPY, BGPUR, BGFILL, BGREAD()**

### 41-1 BGCHK()

Obtains data on the status of the BG screen which has been modified using the BGOFS command.

Format	BGCHK(layer)		
Parameters	Layer	0=Foreground, 1=Rear	
Returns	Number	Interpolation has ended when value reaches 0.	
Error		b00	Coordinate

### 41-2 BGCOPY

This command copies the assigned block area of the BG screen.

Format	BGCOPY layer, start x, start y, end x, end y, transfer destination x, transfer destination y	
Parameters	Layer	0=Foreground, 1=Rear
	x start point	0 – 63
	y start point	0 – 63
	x end point	0 – 63
	y end point	0 – 63
	Transfer destination x	0 – 63
	transfer destination y	0 – 63
Returns	None	
Error		

### 41-3 Screen Data Features

#### ◆Screen Data

This combines the character and palette numbers of the BG screen, as well as data on horizontal and vertical rotation. It is expressed in a 16-bit value.

b00	↑ 10 bit character number (0 – 1023) ↓	
b01		
b02		
b03		
b04		
b05		
b06		
b07		
b08		
b09	↑ 4 bit palette number (0~15) ↓	
b10		Horizontal rotation (0=OFF, 1=ON)
b11		Vertical rotation (0=OFF,1=ON)
b12		
b13		
b14		
b15		

#### ◆Screen Data Character String

The screen data is converted into a 4-digit, hexadecimal number.

(e.g.)

If screen data = &H103F

The screen data character string will be "103F"

#### 41-4 BGPUP (Expert)

##### ◆Writing Screen Data

Format	BGPUP layer, x coordinate, y coordinate, screen data	
Parameters	Layer	0=Foreground, 1=Rear
	x coordinate	0 – 63
	y coordinate	0 – 63
	Screen Data	Hexadecimal 4 – figure value
Returns	None	
Error		

##### ◆Write Screen Data Character String

Format	BGPUP layer, x coordinate, y coordinate, character string	
Parameters	Layer	0=Foreground,

Returns		1=Rear
	x coordinate	0 – 63
	y coordinate	0 – 63
	Strings	4 character strings
	None	
Error		

#### 41-5 BGFILL(Expert)

##### ◆ Fill Screen Data Area

Format	BGP layer, x coordinate, y coordinate, character stringBGFILL layer, start x, start y, end x, end y, screen data	
Parameters	Layer	0=Foreground, 1=Rear
	x start point	0 – 63
	y start point	0 – 63
	x end point	0 – 63
	y end point	0 – 63
	Screen Data	Hexadecimal 4 – figure value
Returns	None	
Error		

##### ◆ Fill Screen Data Area From Character String

Format	BGFILL layer, start x, start y, end x, end y, character string	
Parameters	Layer	0=Foreground, 1=Rear
	x start point	0 – 63
	y start point	0 – 63
	x end point	0 – 63
	y end point	0 – 63
	Strings	4 character strings
Returns	None	
Error		

#### 41-6 BGREAD(Expert)

##### ◆ Obtain Screen Data

Format	BGREAD ( layer, x coordinate, y coordinate ), SC
--------	--

Parameters	Layer	0=Foreground, 1=Rear
	x coordinate	0 – 63
	y coordinate	0 – 63
	SC	Variable to obtain screen data
Returns	None	
Error		

◆Obtain Data Using Screen Data Character String

Format	BGBREAD ( layer, x coordinate, y coordinate ), SC\$	
Parameters	Layer	0=Foreground, 1=Rear
	x coordinate	0 – 63
	y coordinate	0 – 63
	SC\$	Variable to obtain screen data character string
Returns	None	
Error		

## 42 Basic Audio Commands

The following commands relate to playing sound effects and background music. For information on creating music using MML, please see the relevant Expert page:

**BEEP, BGMPLAY, BGMSTOP, BGMCHK(), BGMVOL**

### 42-1 BEEP

Plays a simple warning sound effect.

Format	BEEP [ waveform number [,pitch [,volume [,panpot]]]]	
Parameters	Waveform number	0~69 (if omitted, number is 0)
	Pitch	-8192 plays sound 2 octaves lower, 0 is the original sound, 8192 plays it 2 octaves higher.
	Volume	0=No sound 127=Maximum
	Panpot	0=from left 64=from center 127=from right
Returns	None	
Error		

#### ● How to Calculate Pitch and Scale

An octave is made up of a scale with a resolution of 4096 different elements, so to calculate the pitch (P) of a half tone, you can use  $P=4096/12$ . Calculating pitch using this value will work out as follows:

#### <Changes to Musical Interval>

C = $P*0$

C # = $P*1$

D = $P*2$

D # = $P*3$

E = $P*4$

F = $P*5$

F # = $P*6$

G = $P*7$

G # = $P*8$

A = $P*9$

A # = $P*10$

B =P\*11

#### 42-2 BGMPLAY

Starts playing song. (Up to 8 songs can be played at the same time).

When specifying the volume for a track, the track number cannot be omitted. Track numbers after 128 will play songs you have written yourself.

Format	BGMPLAY [ track number, ] song number [ , track volume ]	
Parameters	Track number	0~7 (Will be 0 unless specified)
	Song number	0~29 128~255
	Track Volume	0~127
Returns	None	
Error		

#### 42-3 BGMSTOP

Stops playing song.

By assigning a Fade Time (measured in seconds), you can cause the music to steadily reduce in volume.

When the track number is omitted, a BEEP command can be used to stop the sound effects and all tracks being played.

Format	BGMSTOP [ track number ] [ , fade time ]	
Parameters	Track number	0~7 (Will be 0 unless specified)
	Fade Time	Time Until end (0=Sudden Stop)
Returns	None	
Error		

#### 42-4 BGMCHK()

This lets you check on current music status.

Format	Variable=BGMCHK( [track number] )	
Parameters	Track number	0~7 (Will be 0 unless specified)
Returns	Number	FALSE=Music stopped TRUE=Music playing
Error		

#### 42-5 BGMVOL

This adjusts the volume for each track.

Format	BGMVOL [ track volume, ] volume	
Parameters	Track number	0~7 (Will be 0 unless specified)
	Track Volume	0 – 127
Returns	None	
Error		

### 43 Audio Commands (Expert)

The following commands are for use when creating music using MML, or exchanging MML data:

**BGMSET, BGMSETD, BGMCLEAR, BGMPPLAY, BGMSETV, BGMGETV(), BGMPRG**

#### 43-1 BGMSET

This saves song data created using MML.

Format	BGMSET song number, MML character string [ , MML character string 2... ]	
Parameters	Song number	128 – 255
	MML character string	This describes song data as a character string.
	MML character string 2	If song data cannot all be expressed in a single character string, it can be divided into multiple strings.
Returns	None	
Error		

#### 43-2 BGMSETD

This saves pre-set MML data.

**When describing MML data in a string, you need to use the expression DATA 0 to signify the end of the MML data.**

Format	BGMSETD song number, @label BGMSETD song number, variable\$	
Parameters	Song number	128 – 255
	@label	Label name specifying data saved in MML form.
Returns	None	
Error		
e,g,	@MMLDATA DATA ""CDEFGAB <C> "" DATA ""CCCEEEGGG"" DATA 0 '---	



	BGMSETD 128, @MMLDATA BGMPLAY 128
--	--------------------------------------

#### 43-3 BGMCLEAR

This clears saved song data.

Format	BGMCLEAR [ song number ]	
Parameters	Song number	128 – 255 (if not specified, will include all song data)
Returns	None	
Error		

#### 43-4 BGMPLAY(Expert)

Use MML directly and play music.

When playing a track using MML, the track will be defined with a 0. The track's volume cannot be adjusted.

(If a track between 1- 7 is assigned, it will result in an error)

Format	BGMPLAY MML character string [ ,MML character string 2... ]	
Parameters	MML character string	Song data is described as a character string.
	MML character string 2	If song data cannot all be expressed in a single character string, it can be divided into multiple strings.
Returns	None	
Error		

#### 43-5 BGMSETV

This command writes to variables within MML.

Format	BGMSETV track number, variable number, variable	
Parameters	Track number	0 - 7
	Variable number	0 – 7 (MML variable \$0 - \$7)
	Value	0 - 255
Returns	None	
Error		

#### 43-6 BGMGETV()

Read variables in MML.

The cycle in which replacement variables within MML will be reflected in the program lasts for 1/60th of a second. Be sure to leave at least a period of VSYNC 1.

Format	Variable=BGMGETV( track number, variable number )	
Parameters	Track number	0 - 7
	Variable number	0 - 7 (MML variable \$0 - \$7)
Returns	Number	-1 = Track Currently Stopped
Error		

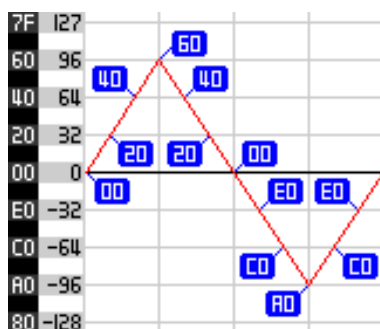
#### 43-7 BGMPRG

Save waveforms of instruments that can be used with MML.

For more information on the A, D, S, R commands used to define the waveform envelope, see MML Commands@E.

Format	BGMPRG instrument number, [ key, ] [ A, D, S, R, ] waveform character string	
Parameters	Instrument number	224 - 255
	A Attack	0 - 127
	D Decay	0 - 127
	S Sustain	0 - 127
	R Release (Until Disappears)	0 - 127
	Waveform character string	Hexadecimal character string (requires 64 or 128 characters)
Returns	None	
Error		

Waveform character strings are 8-bit values displayed in two-digit hexadecimal form.



If the digits inside the blue frame above are converted into waveform character strings, they will appear like this:

"00204060402000E0C0A0C0E0"

In this way, 64-character waveform strings will be treated as the waveform for a single cycle. A 128-character string can be used to define a more complex waveform. While complex waveforms used to sample real instruments cannot be saved, it can be used as a basic synthesizer.

## 44 DS Wireless Play


### ■ What you will need





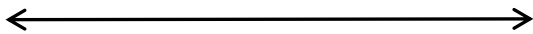
- Two Nintendo DSi systems on which Petit Computer has been saved.

### <Procedure>

1. Turn the system's power ON.
2. Touch Petit Computer on the DSi Menu Screen.
3. When sending or receiving data, select "File Management" from the Home Menu and then touch either the "Send" or "Receive" button. Alternatively, after launching BASIC, you can use the SENDFILE command to transmit resources and the RECVFILE command to receive them.
4. For more information about files, resources and the transmission and receipt of data, please see "13. Files and Resources".

### Important Wireless Communication Guidelines

During wireless game play, an icon () will appear on either the upper or lower screen showing the strength of the wireless signal. The icon has four modes depending on the signal strength, as shown below.

				
Number of Bars	0	1	2	3
Signal Strength				
	Weaker		Stronger	

Begin with the distance between systems at about 30 feet or less and move closer or farther apart as desired, keeping the signal strength at two or more bars for best results.

Keep the maximum distance between systems at 65 feet or less.

The systems should face each other as directly as possible.

Avoid having people or other obstructions between the DS systems.

Avoid interference from other devices. If communication seems to be affected by other devices (wireless LAN, microwave ovens, cordless devices, computers), move to another location or turn off the interfering device.

To play wirelessly, you must first set wireless Communications to ON in the Nintendo DSi System settings.

